L3. Advanced LLM topics: Multimodality, tool usage and multi-agents

James Zou

April 7, 2025

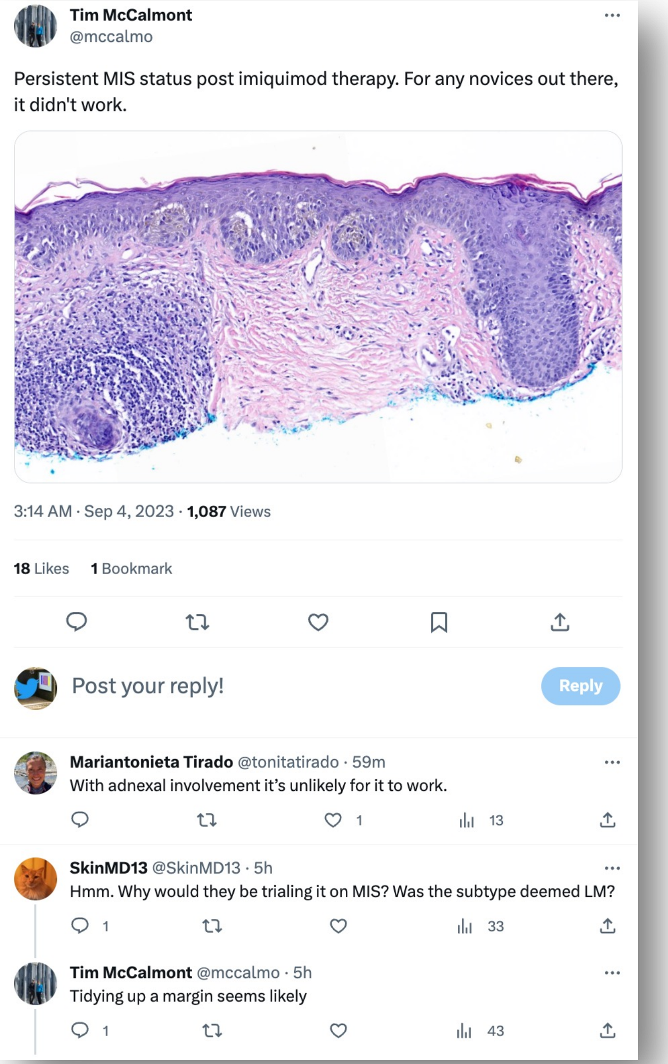
www.james-zou.com

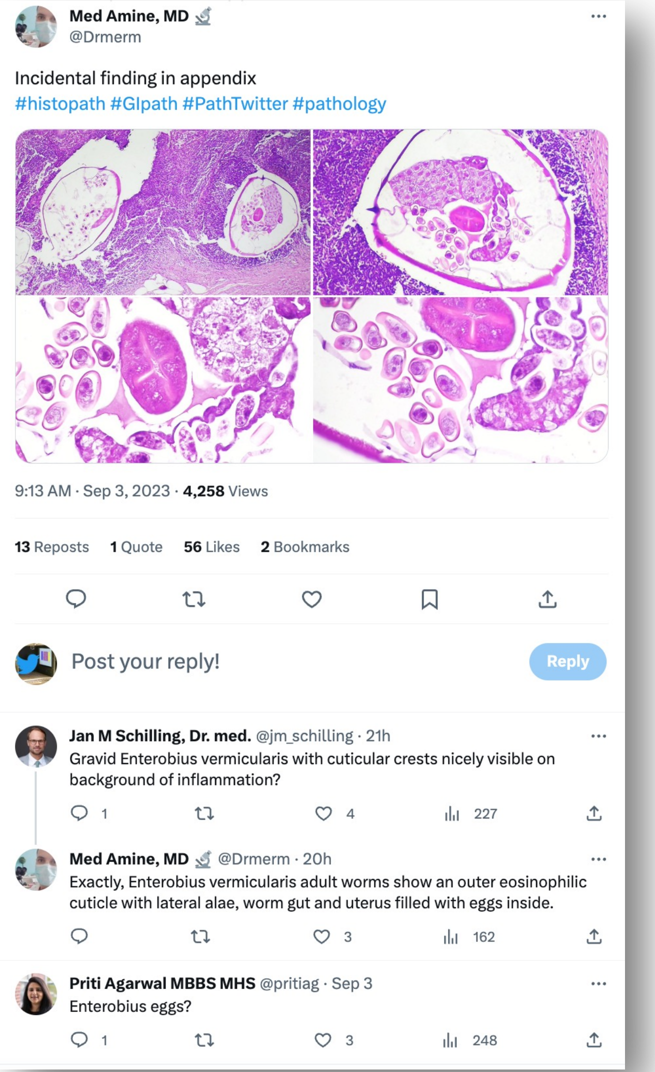
jamesz@stanford.edu

Pathology is core to diagnosis and treatment planning

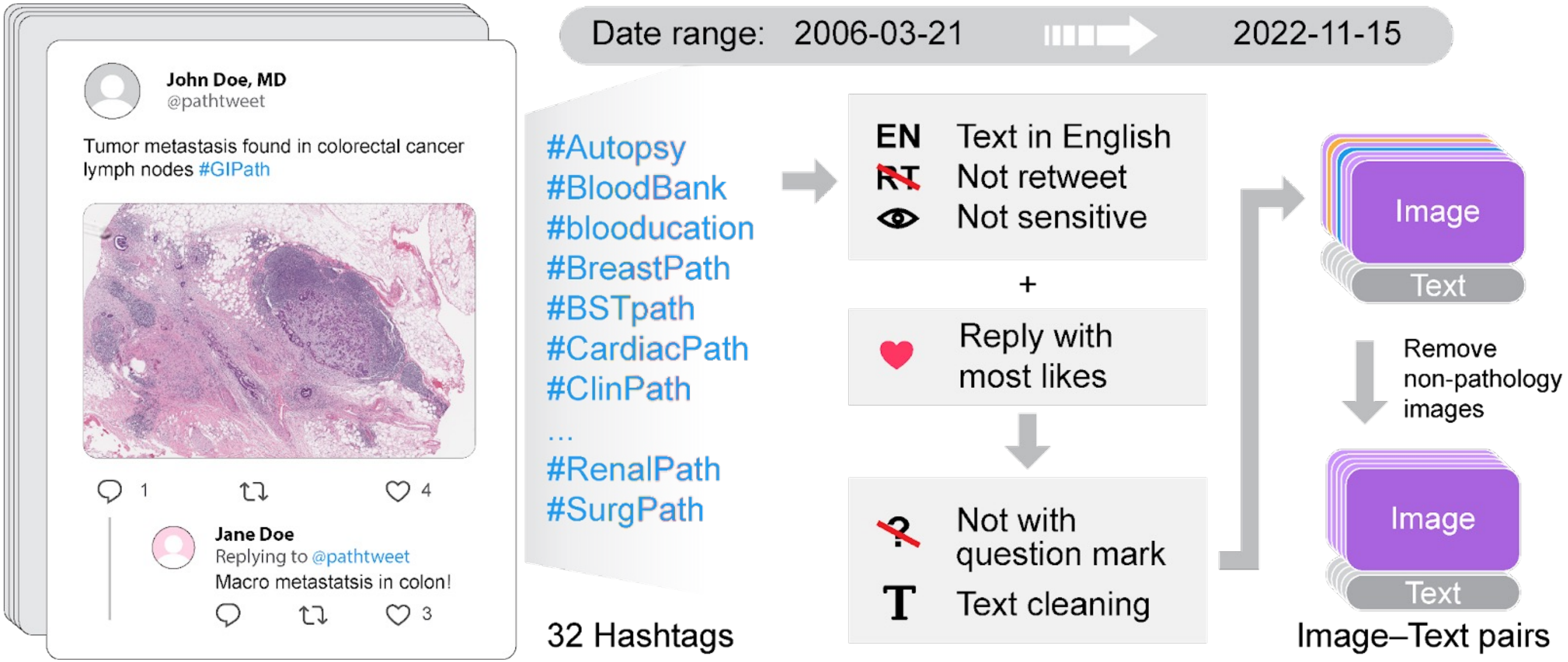
Pathologists often encounter images that they are not familiar with.

Much medical knowledge is shared on Twitter





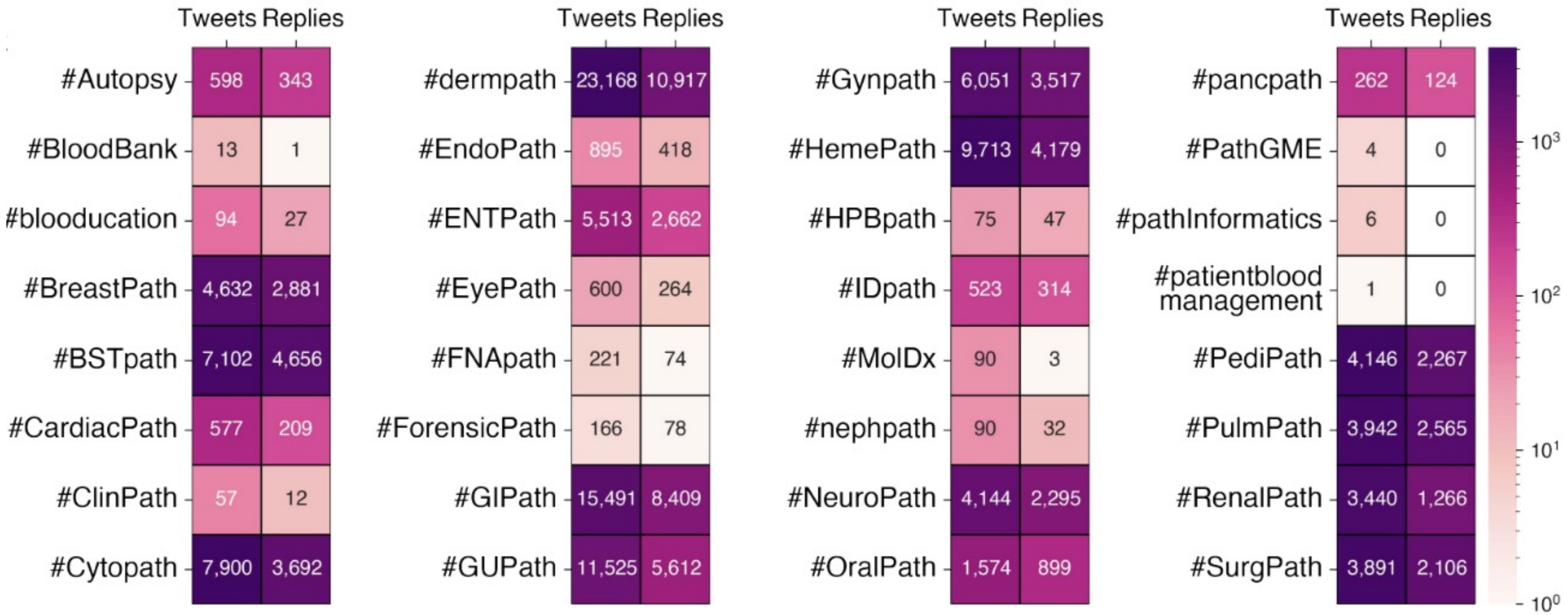
Creating OpenPath: >200K high-quality Twitter image-text pairs



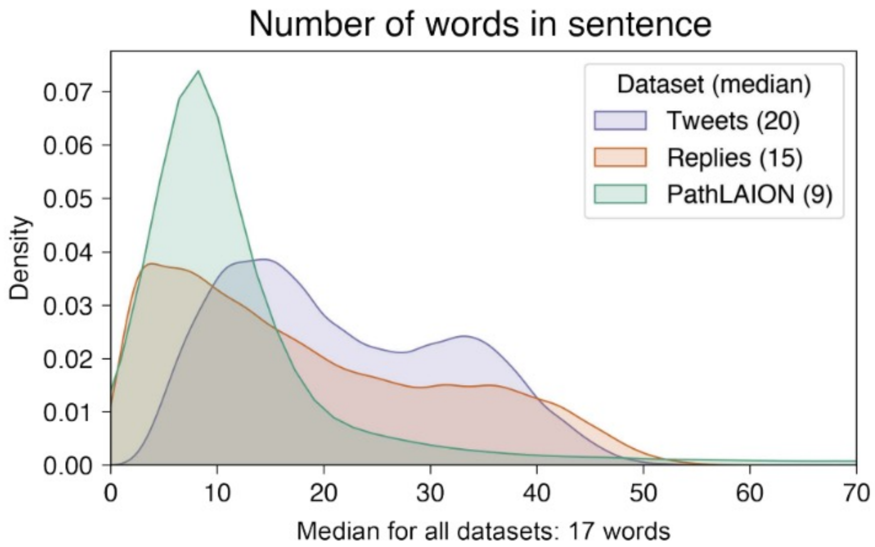
Largest public dataset of pathology image + discussions.

Huang*, Bianchi* et al. *Nature Medicine* 2023.

Creating OpenPath: >200K high-quality Twitter image-text pairs

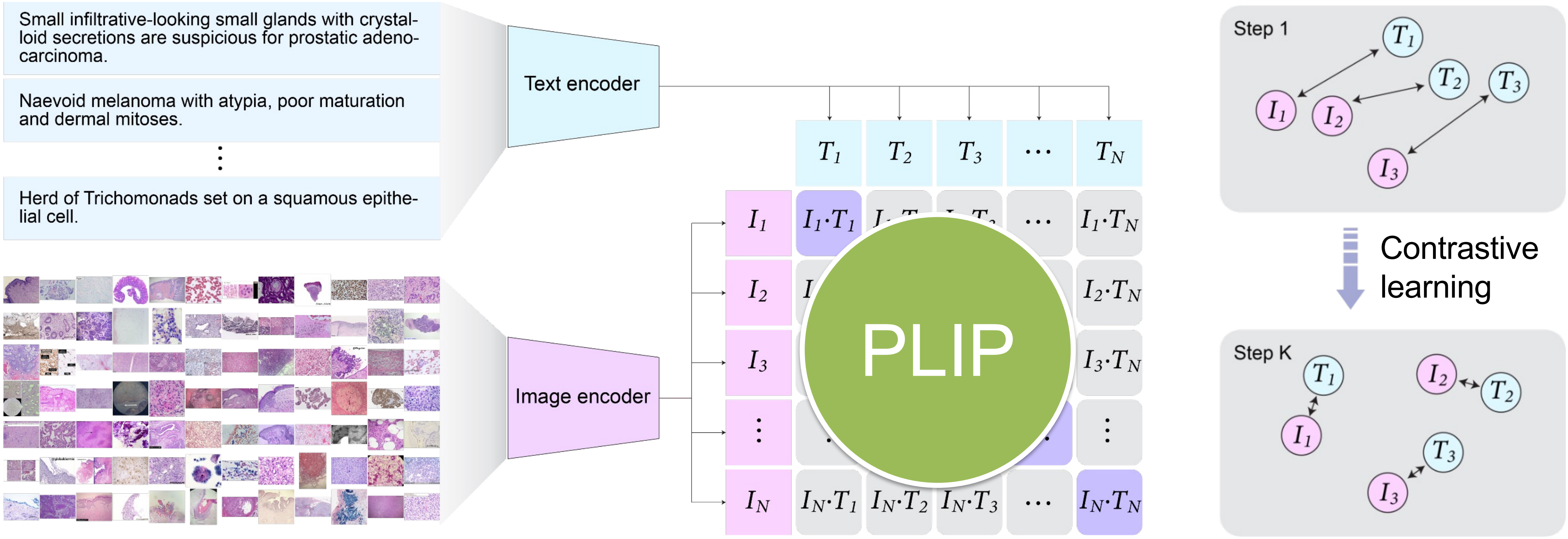


# of Tweets in each sub-area of pathology



Median text length = 20 words.

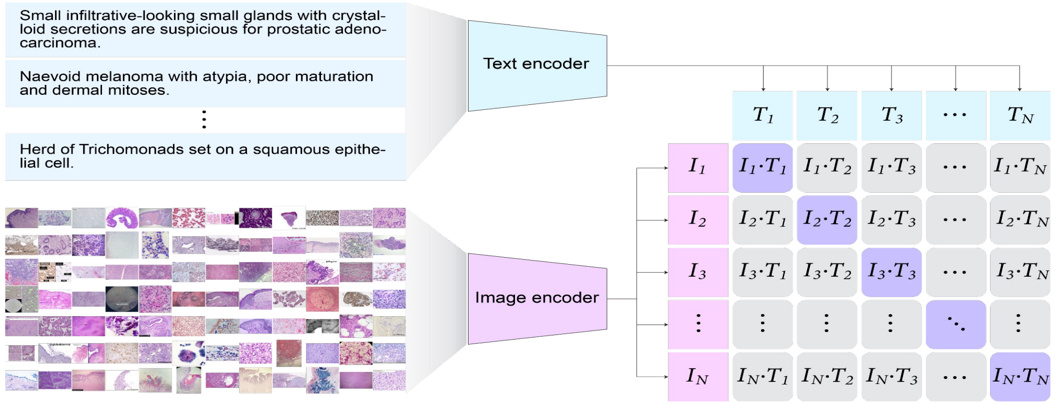
Using Twitter to train the largest visual-language AI for pathology



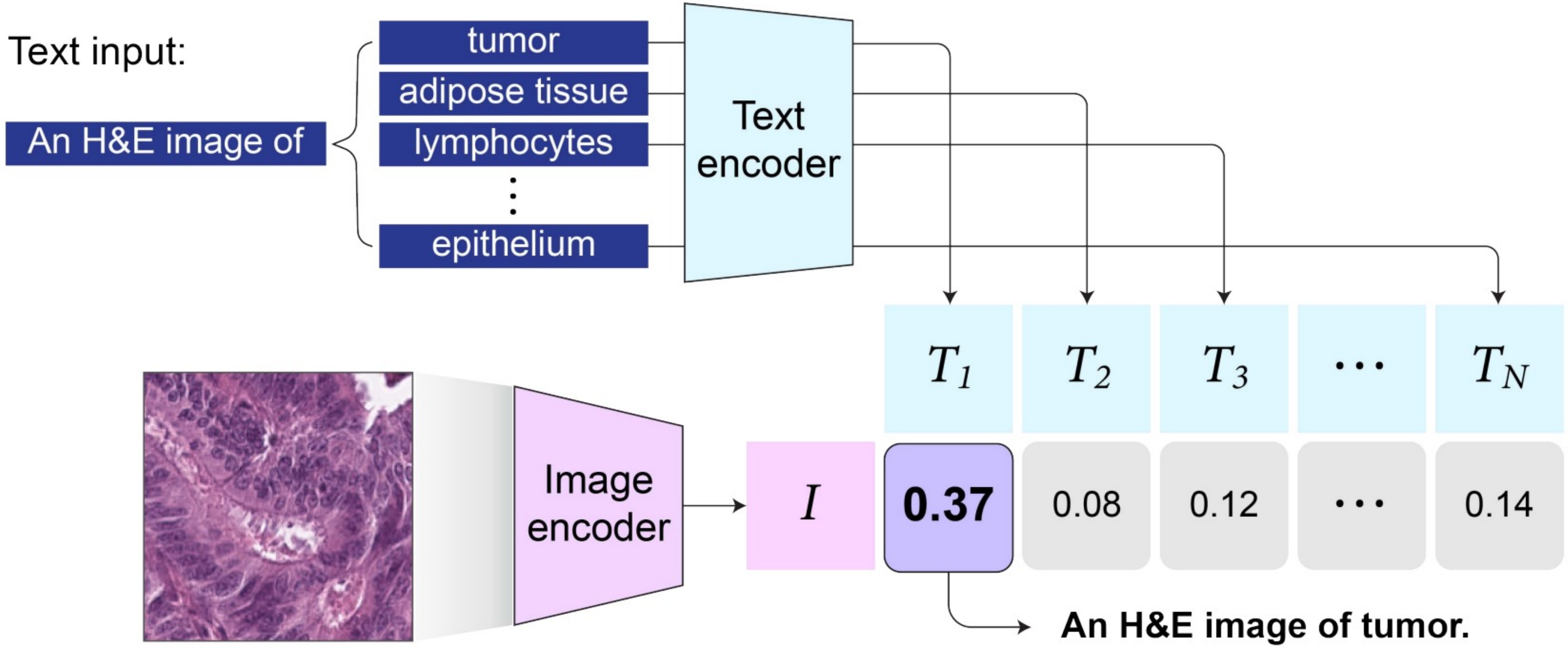
Huang*, Bianchi* et al. *Nature Medicine* 2023.

PLIP applications

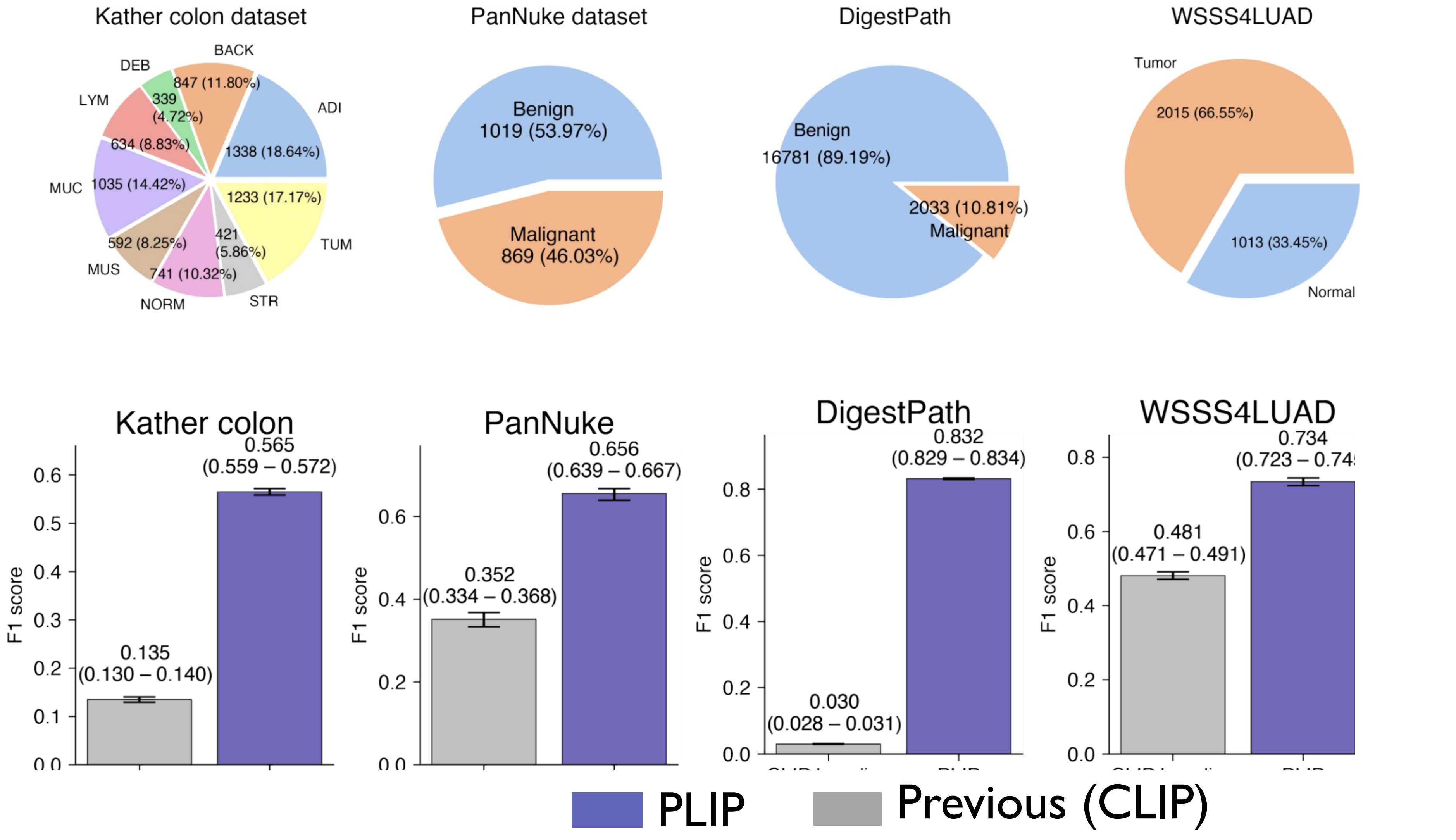
1. Zero-shot classification.
2. Improved representation learning for downstream tasks.
3. Text-to-image retrieval
4. Image-to-image retrieval



1. PLIP can perform zero-shot classification on new images



PLIP can perform zero-shot classification on new images

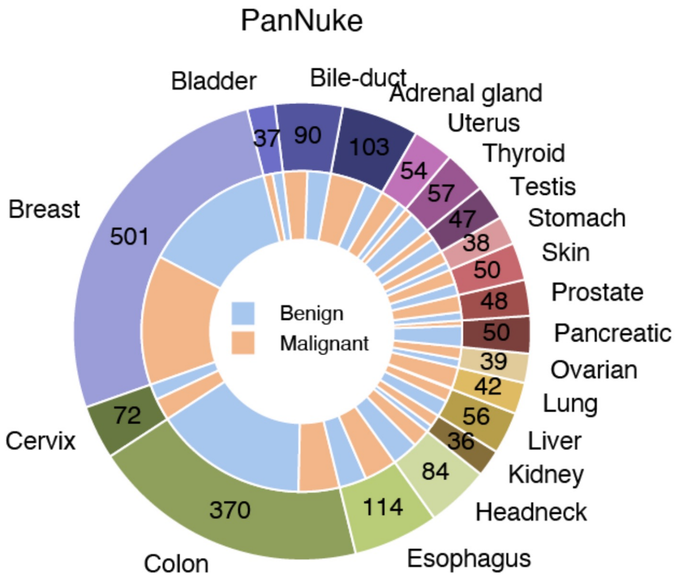


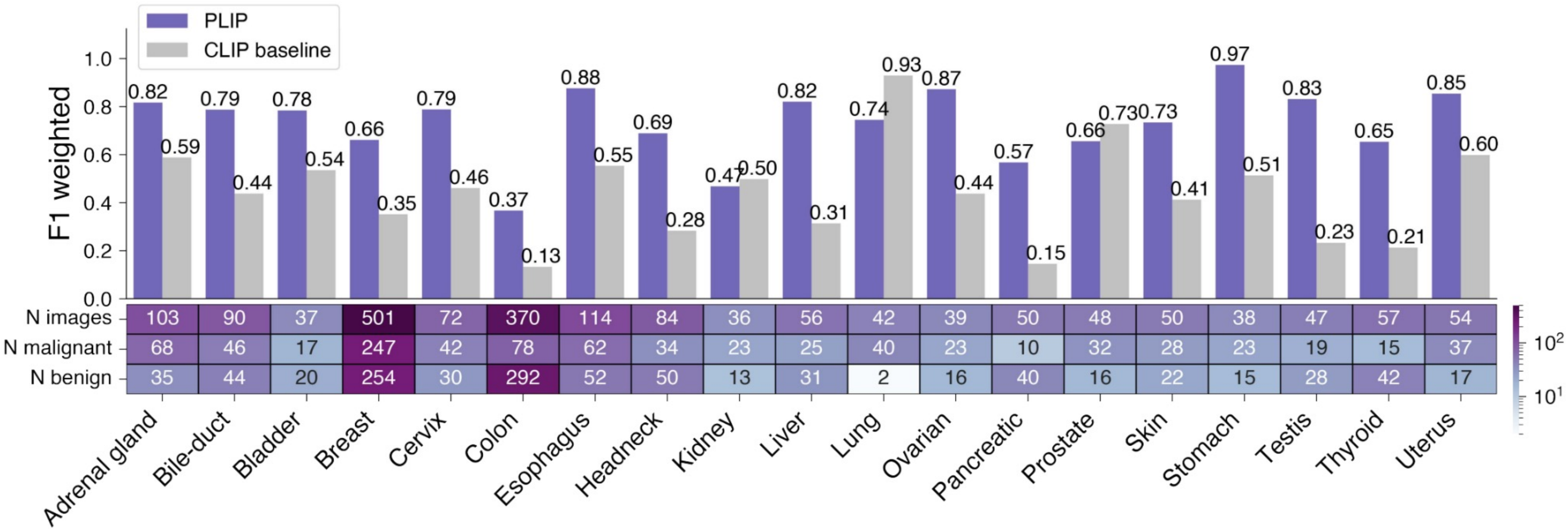
External validation datasets:

- Kather colon
- PanNuke
- DigestPath
- WSSS4LUAD

Metric: F1 score (ranges 0-1, higher the better)

PLIP can perform zero-shot classification on new images

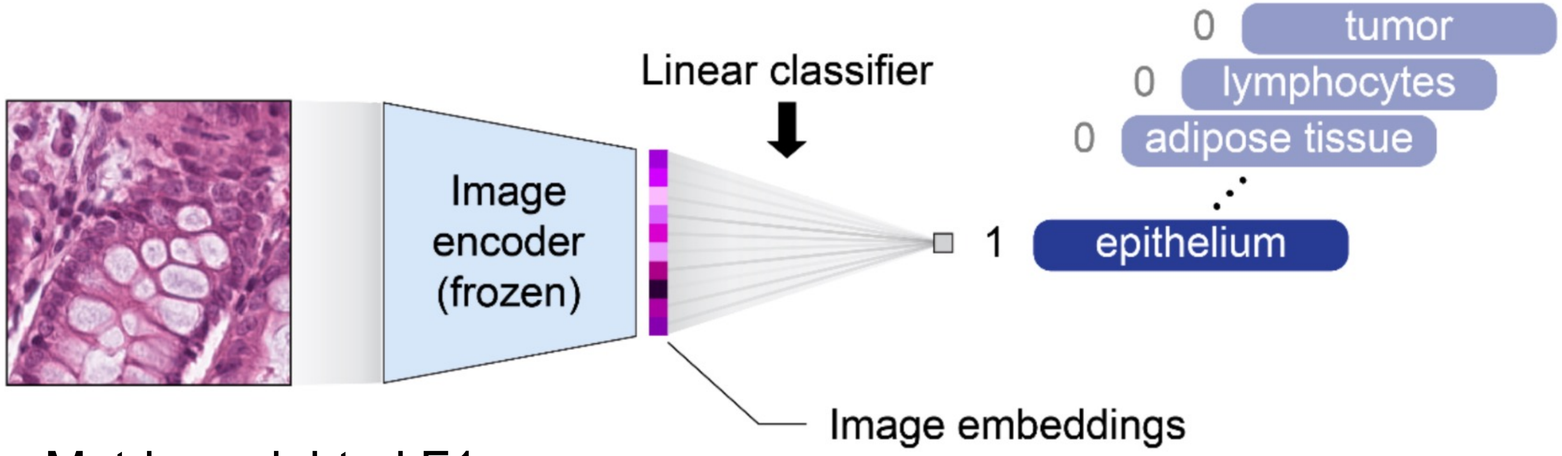




PLIP achieved superior F1 score on 16 out of 19 organs.

- 7 organs achieved reasonably high F1 scores (> 0.8).
- While the baseline CLIP performed only at F1 = 0.3 ~ 0.6.

2. PLIP provides better image representations for training models



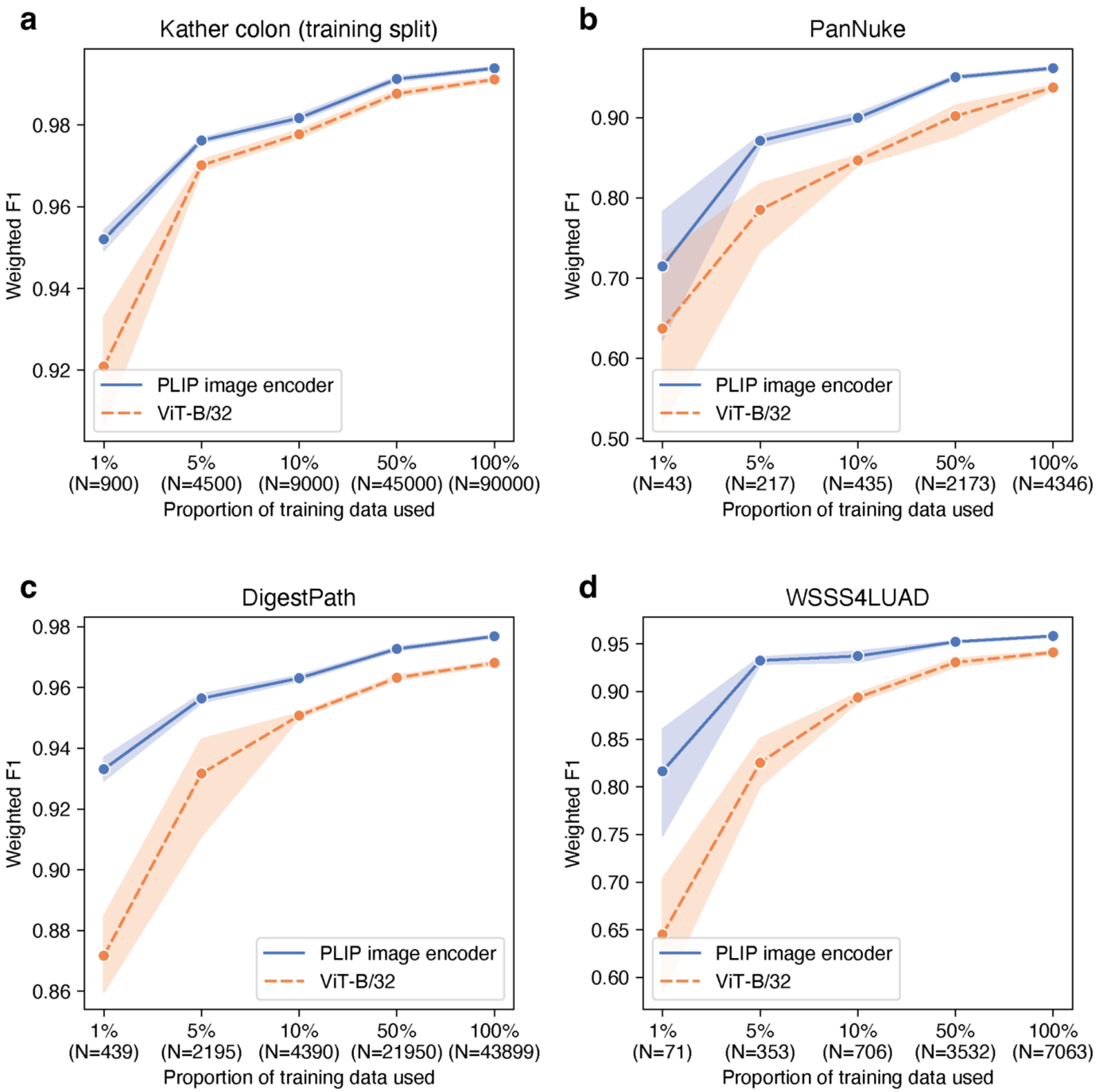
Use cases:

- Fine-tune model for a task-specific problem.

Metric: weighted F1 score

	Kather colon	PanNuke	DigestPath	WSSS4LUAD	Average
CLIP	0.797 (±0.006)	0.852 (±0.002)	0.753 (±0.009)	0.850 (±0.022)	0.813 (±0.043)
MuDiPath	0.825 (±0.001)	0.896 (±0.001)	0.827 (±0.007)	0.917 (±0.003)	0.866 (±0.041)
PLIP	**0.877 (±0.001)**	**0.902 (±0.010)**	**0.856 (±0.008)**	**0.927 (±0.007)**	**0.891 (±0.028)**
PLIP vs. CLIP	2.9e-9	9.4e-6	1.5e-7	1.5e-4	—
PLIP vs. MuDiPath	9.4e-12	0.249	6.2e-4	3.0e-2	—

2. PLIP provides better image representations for training models

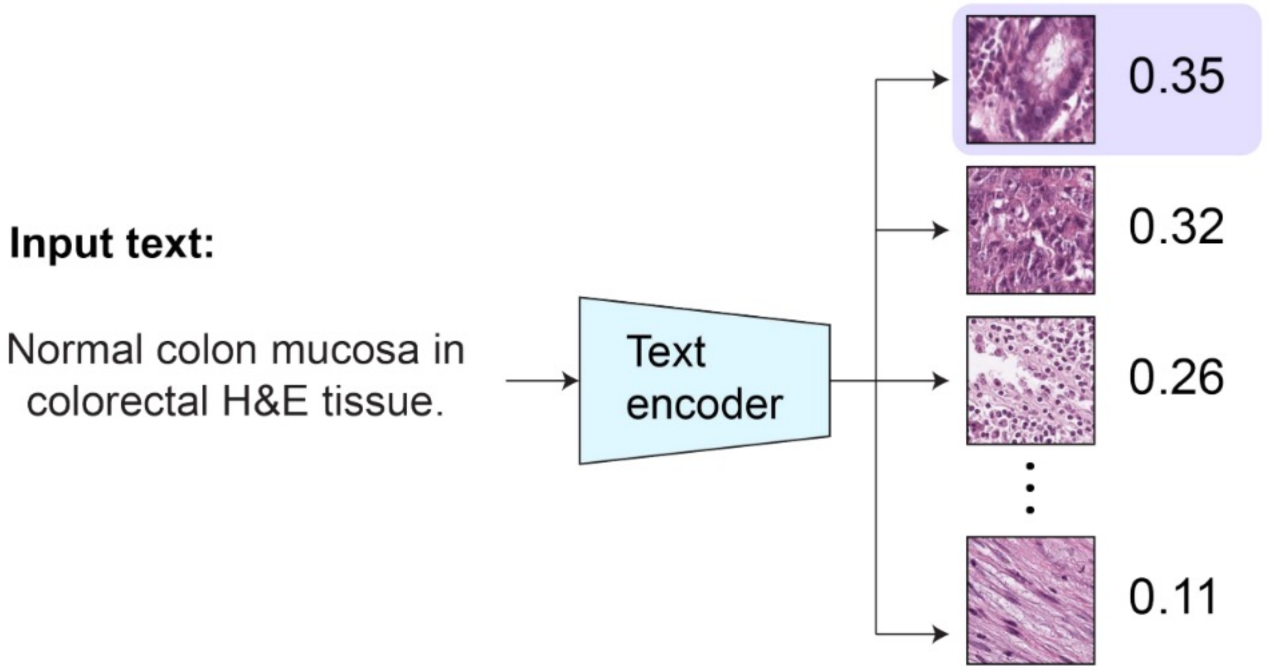


Compared to end-to-end supervised learning algorithm ViT-B/32.

- PLIP achieves better performances.
- The improvement is especially large when the training set size is small.

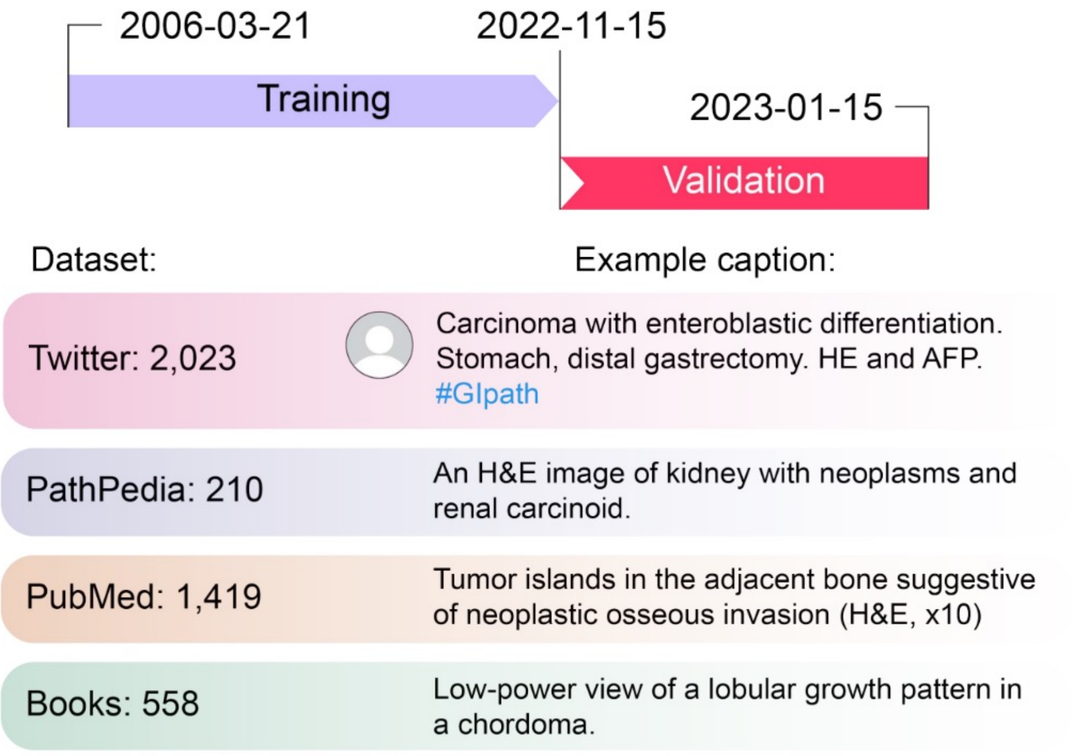
3. Text-to-image retrieval

Paradigm:



Candidate pool

Evaluation datasets:



3. Text-to-image retrieval

Dataset	Number of candidates	Metric	PLIP	CLIP	Random	Fold change (PLIP vs. CLIP)
Twitter	2,023	Precision@10	0.271	0.061	0.005	4.5
		Precision@50	0.527	0.128	0.025	4.1
PathPedia	210	Precision@10	0.409	0.167	0.048	2.5
		Precision@50	0.752	0.476	0.238	1.6
PubMed	1,419	Precision@10	0.069	0.015	0.007	4.7
		Precision@50	0.206	0.082	0.035	2.5
Books	558	Precision@10	0.265	0.045	0.018	5.9
		Precision@50	0.659	0.165	0.090	4.0

0 5 10 15 20 >25

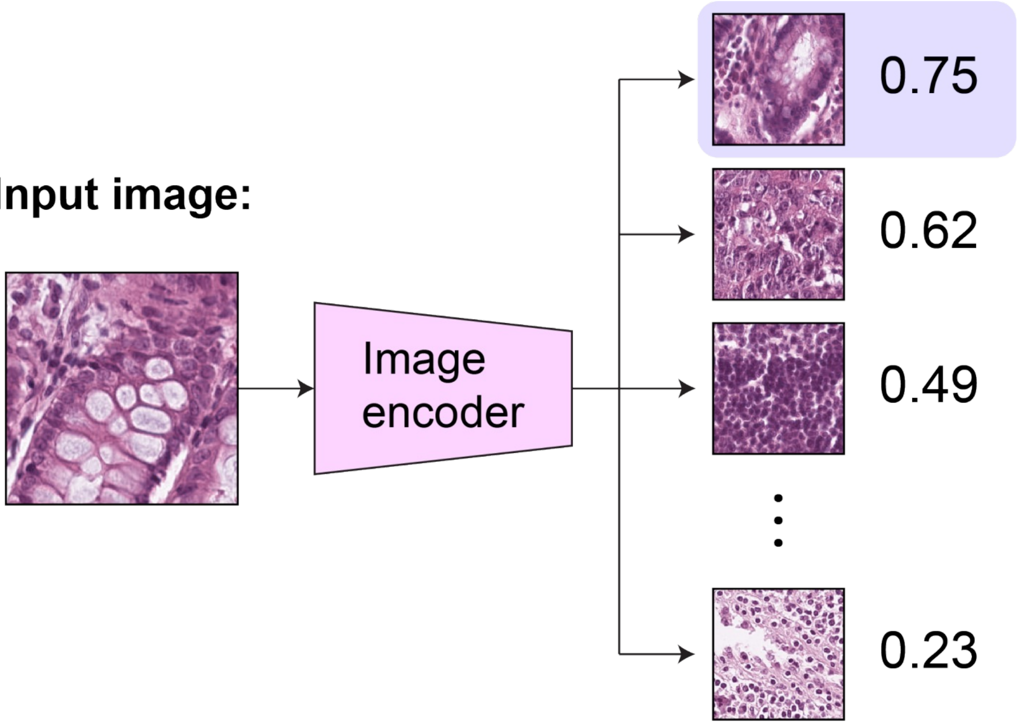
Fold change compared to random performances

0 1 2 3 4 5 6

Fold change compared to CLIP performances

4. Image-to-image retrieval

Input image:

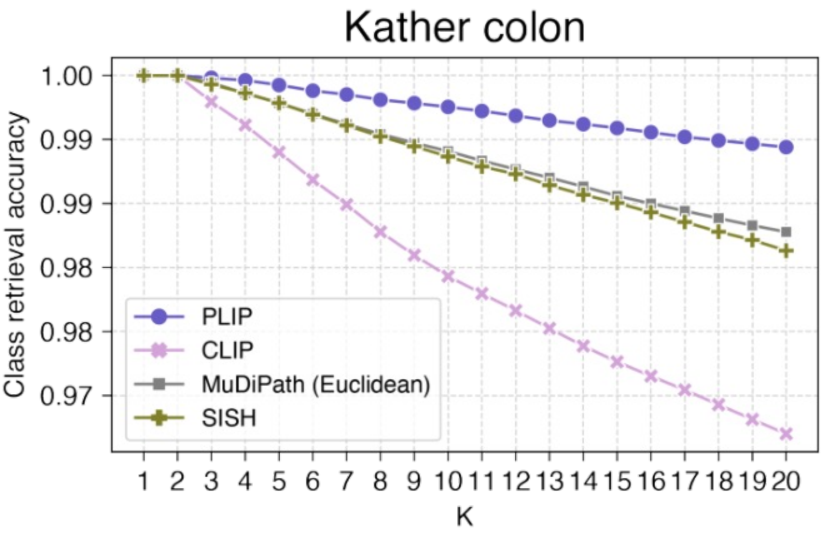


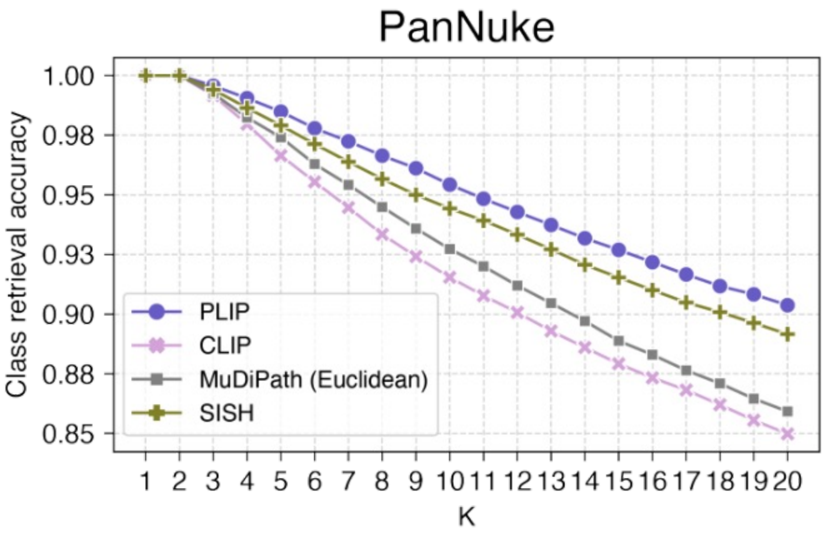
Candidate pool

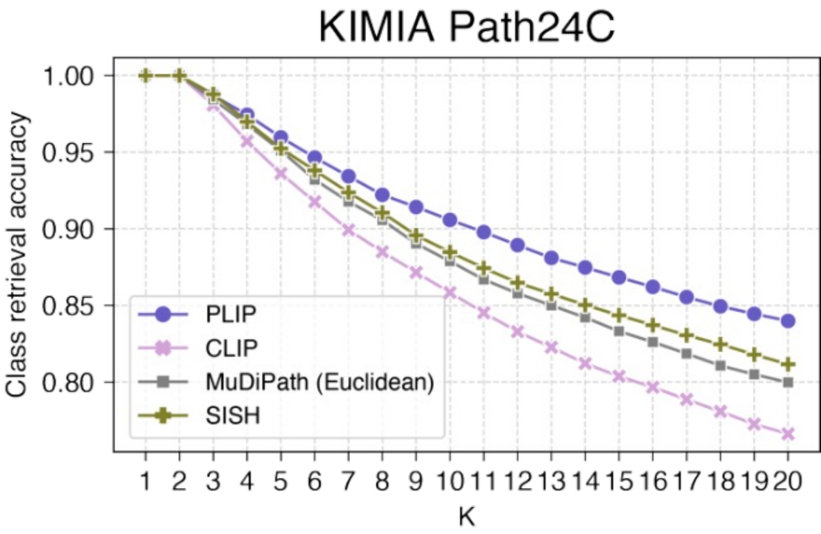
Use cases:

- Search similar images
- Education
- Second opinion
- ……

PLIP retrieves clinically relevant cases







9 tissue types

19 organ types

24 staining styles

Web demo

👈 Search engine is available online now.

PLIP can serve as a powerful search engine for medicine

Text-to-image retrieval:

pyramidal neuron in hippocampus

Submit

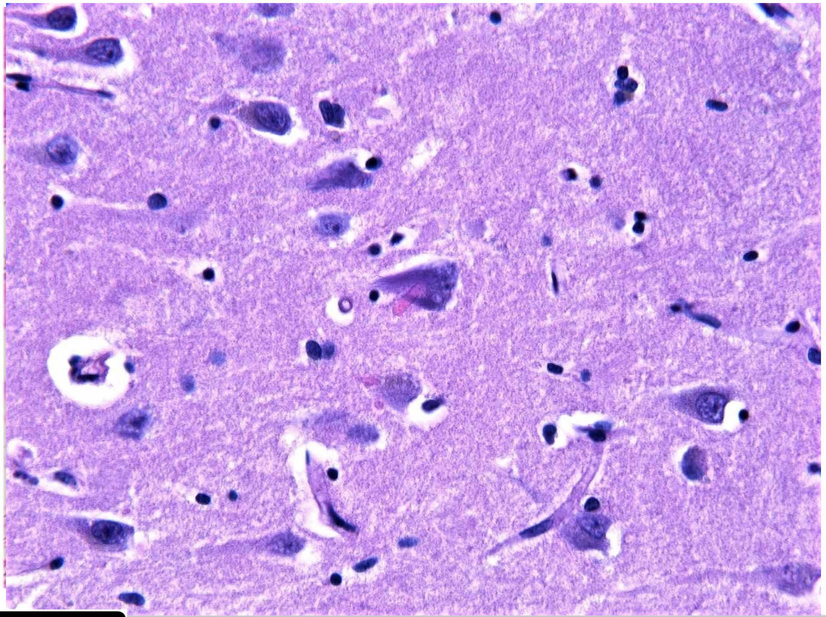
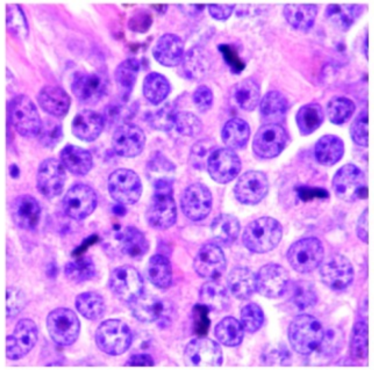


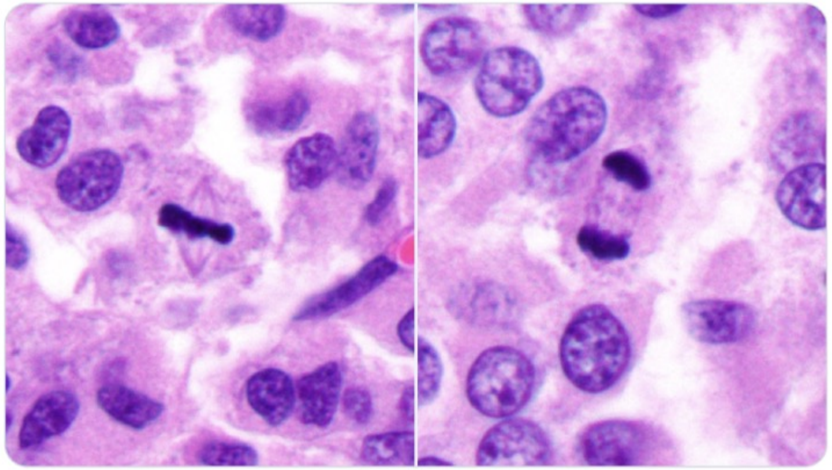
Image-to-image retrieval:

Drop file here Browse files



Input image: mitotic figure

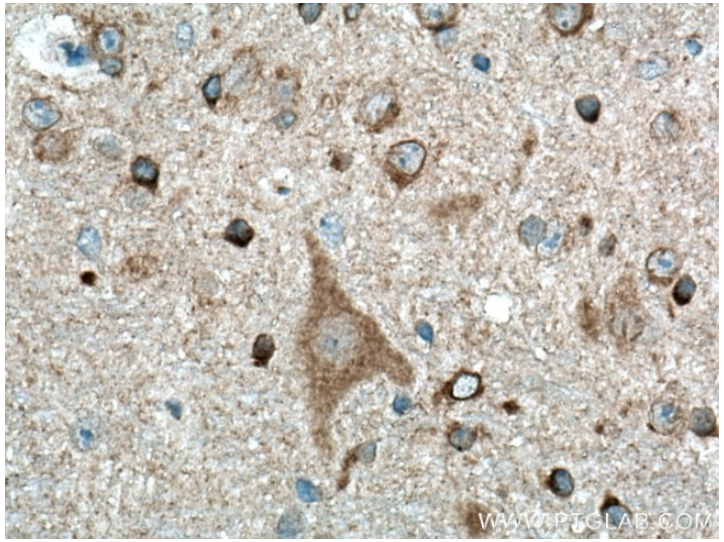
Most relevant image (similarity = 0.9091):



Web demo

👈 Search engine is available online now.

Example: retrieving amyloid beta IHC

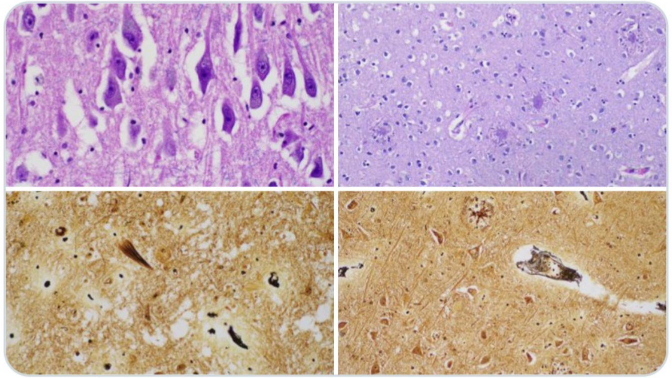


Input image

Top result:

Misha Movahed-Ezazi, MD
@MishiePishi · **Follow**

Elderly patient with plaque and tangle pathology. Classic granulovacuolar changes in neurons of subsector CA1 of the hippocampus. Numerous neuritic plaques and tangles. #pathology #PathTwitter #MedTwitter #Neuropathology #Neuropath #Autopsy #forensics #Neurology #Neurosurgery



12:06 PM · Jul 25, 2022

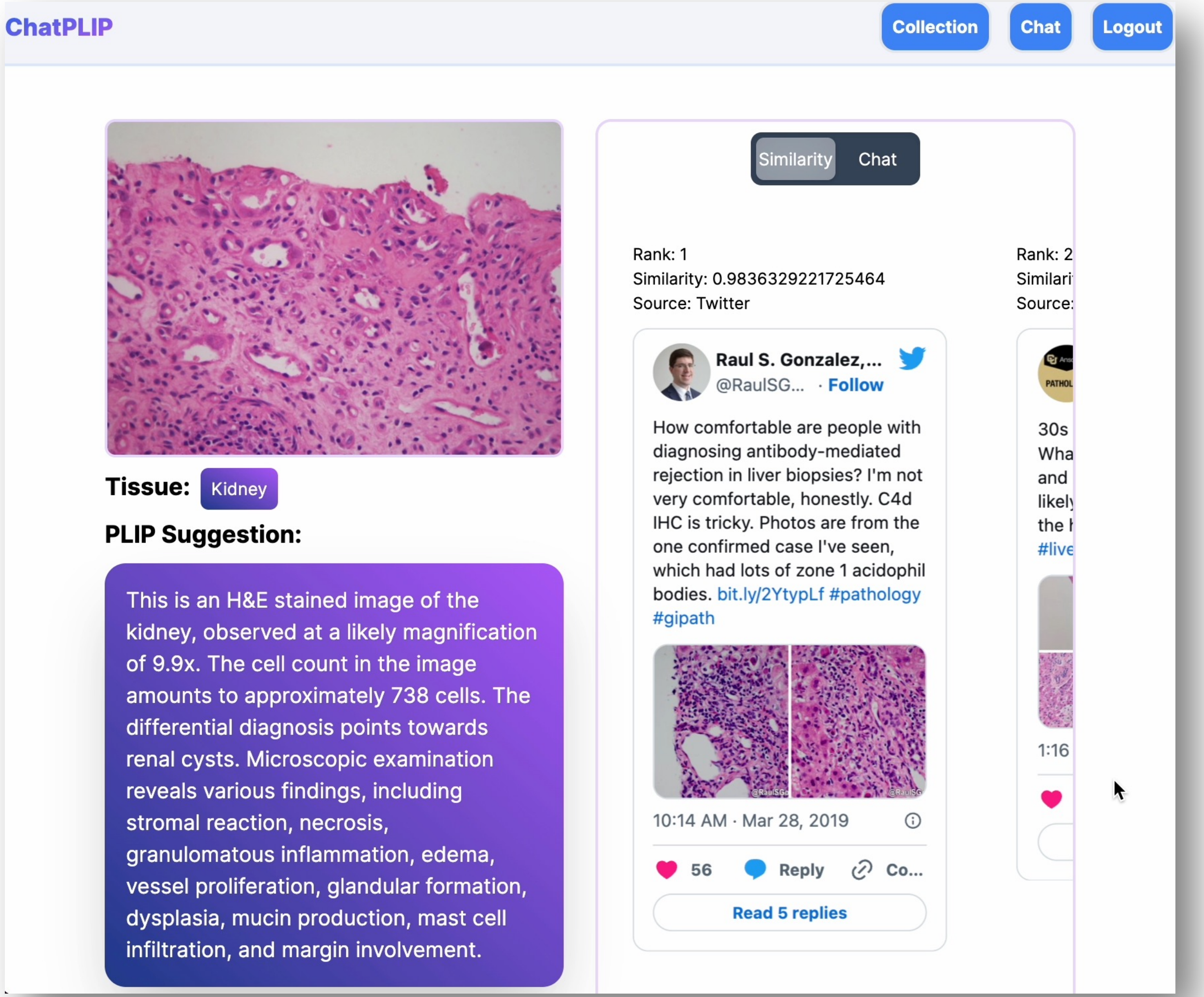
26 Reply Copy link

Read more on Twitter

Web demo

👉 Search engine is available online now.

PLIP facilitates knowledge sharing and clinical decision support.



Huang*, Bianchi* et al. *Nature Medicine* (cover story) 2023.

Augmenting LLM with tools

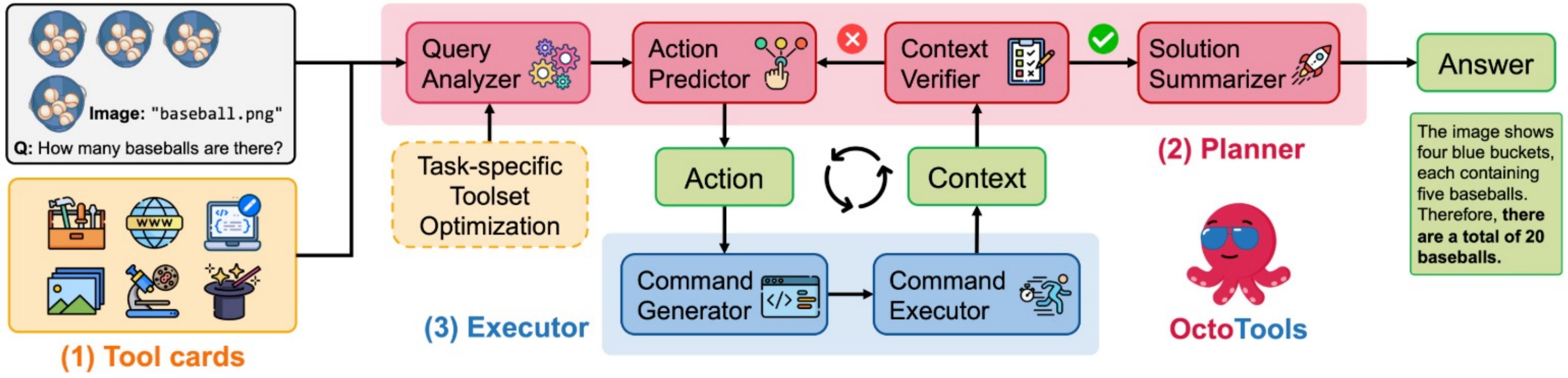
OctoTools

An Agentic Framework with Extensible Tools for Complex Reasoning

Pan Lu*, Bowen Chen*, Sheng Liu*, Rahul Thapa, Joseph Boen, James Zou
Stanford University
* Equal Contribution

Paper | arXiv | Code | Demo | Visualize | Twitter | Discord

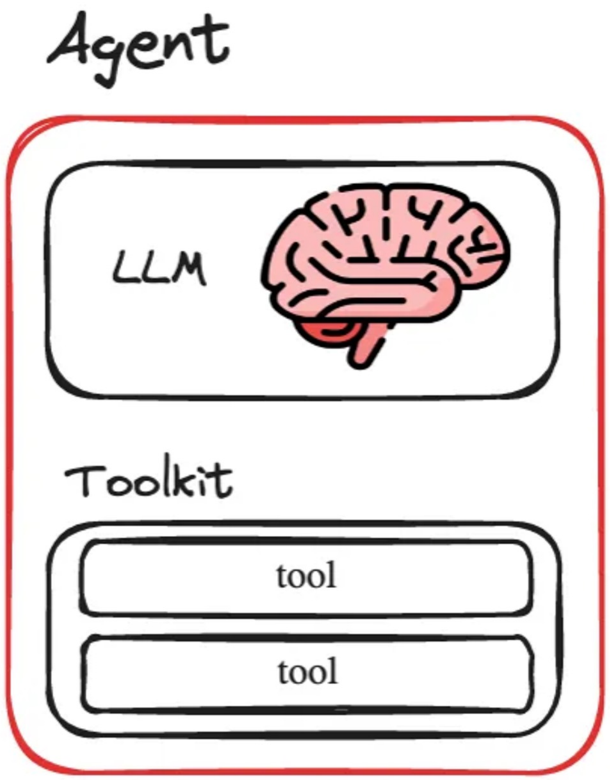
https://octotools.github.io/



Pan Lu, Bowen Chen, Sheng Liu et al 2025

Questions We Want to Address

- **To build an agent system**
    - Train-free, user-friendly, scalable
    - General purposes
    - Domain-specific applications
- **How the agent system behave in complex reasoning**
    - Performance
    - Tool usage
    - Average steps v.s. maximal allowed steps
    - Full toolset v.s. Optimized toolset
    - Cost analysis



Our Solution: OctoTools!

We propose **OctoTools**, an open-source, versatile, and user-friendly agent-toolbox framework for complex reasoning tasks.

- Flexible task planning
- Multi-step problem solving
- Effective tool usage

- Comprehensive experiments
- Consistent performance gains
- In-depth study

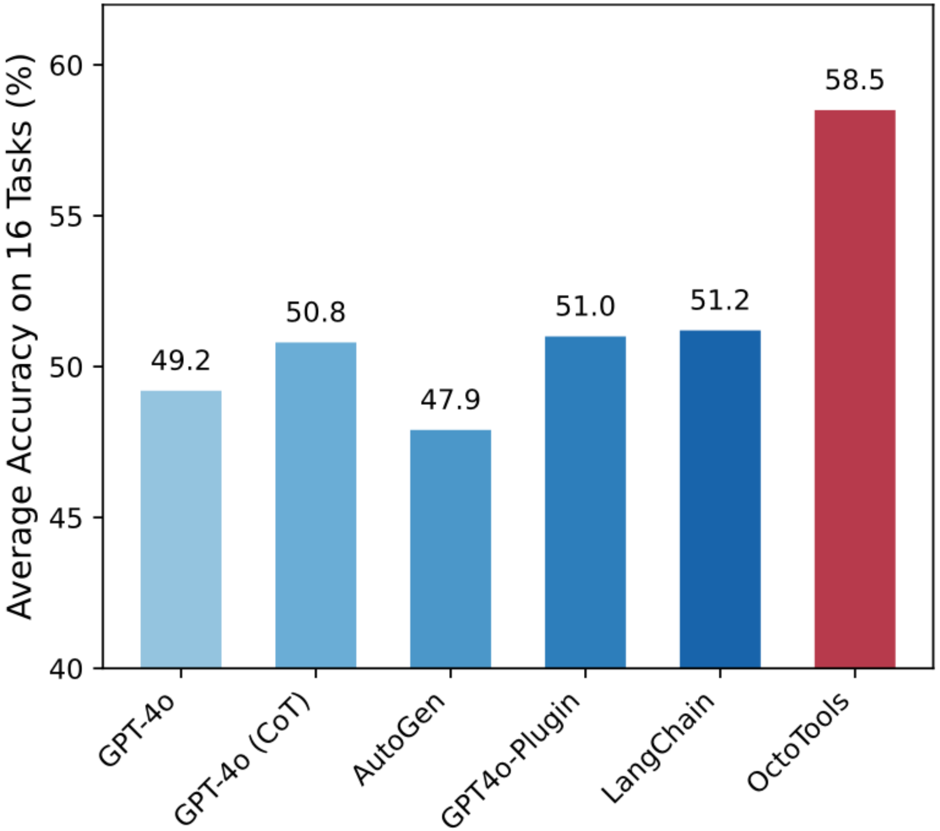
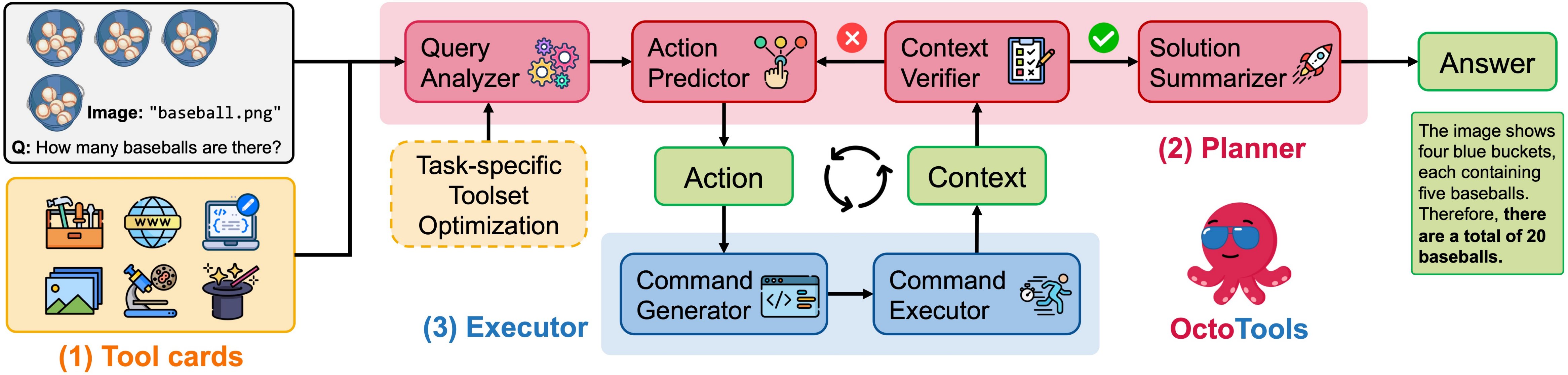


Figure 1. Performance comparison across 16 benchmarks. On average, our OctoTools system achieves an average accuracy gain of 9.3% over GPT-4o and 7.3% over LangChain.

The OctoTools Framework



Tool cards: define tool-usage metadata, encapsulate heterogeneous tools

Planner: govern both high-level and low-level planning to address the global objective, refine actions step by step

Executor: instantiate tool calls by generating executable commands, save structured results in the context

Tool Cards

https://octotools.github.io/#tool-cards

Image Captioner Tool: Metadata

```
tool_name="Image_Captioner_Tool",

tool_description="A tool that generates captions for images using OpenAI's multimodal
    model.",

input_types={
   "image": "str - The path to the image file.",
   "prompt": "str - The prompt to guide the image captioning (default: 'Describe this
       image in detail.').",
},

output_type="str - The generated caption for the image.",

demo_commands=[
    {
      "command": 'execution = tool.execute(image="path/to/image.png")',
      "description": "Generate a caption for an image using the default prompt and
          model."
    },
    {
      "command": 'execution = tool.execute(image="path/to/image.png", prompt="Explain
          the mood of this scene.")',
      "description": "Generate a caption focusing on the mood using a specific prompt
          and model."
    }
],

user_metadata = {
   "limitation": "The Image_Captioner_Tool provides general image descriptions but has
       limitations: 1) May make mistakes in complex scenes, counting, attribute
       detection, and understanding object relationships. 2) Might not generate
       comprehensive captions, especially for images with multiple objects or abstract
       concepts. 3) Performance varies with image complexity. 4) Struggles with
       culturally specific or domain-specific content. 5) May overlook details or
       misinterpret object relationships. For precise descriptions, consider: using it
       with other tools for context/verification, as an initial step before refinement,
        or in multi-step processes for ambiguity resolution. Verify critical
       information with specialized tools or human expertise when necessary."
}
```

Generalist Solutioner
Base tool that answers general questions without using any external tools.

Metadata | Code | Example

Image Captioner
Generate a caption for a given image with a text prompt.

Metadata | Code | Example

Relevant Patch Zoomer
Locate and zoom in relevant quarter patches in an image given a question.

Metadata | Code | Example

Text Detector
Detect text with coordinates and confidence scores in an image by EasyOCR.

Metadata | Code | Example

Object Detector
Detect objects in an image using the Grounding DINO model.

Metadata | Code | Example

Wikipedia Search
Search Wikipedia for relevant information based on a given query.

Metadata | Code | Example

Google Search
Search the Google website for relevant information based on a given query.

Metadata | Code | Example

URL Extractor
Visit the given URL and extract all text from that page.

Metadata | Code | Example

Python Interpreter
Generate and execute Python code snippets for basic calculations.

Metadata | Code | Example

ArXiv Paper Search
Search arXiv for the latest literature based on a given query.

Metadata | Code | Example

PubMed Paper Search
Search PubMed for the latest literature based on a given query.

Metadata | Code | Example

Nature News Search
Search the latest news articles from the Nature website.

Metadata | Code | Example

Pathology Classifier
Classify H&E-stained pathology images into one of the given options.

Metadata | Code | Example

More Tools ...
More tools can be added!

Metadata | Code | Example

Extendable Toolbox

Google_Search
Wikipedia_Knowledge_Searcher
Arxiv_Paper_Searcher
url_text_extractor

Python_Code_Generator

Image_Captioner
Text_Detector
Relevant_Patch_Zoomer
Advanced_Object_Detector

Generalist_Solution_Generator

Path_Generalist_Classifier
Pubmed_Search

pubmed_searcher
nature_news_fetcher

human_phenotype_ontology
medical_action_ontology
mondo_disease_ontology

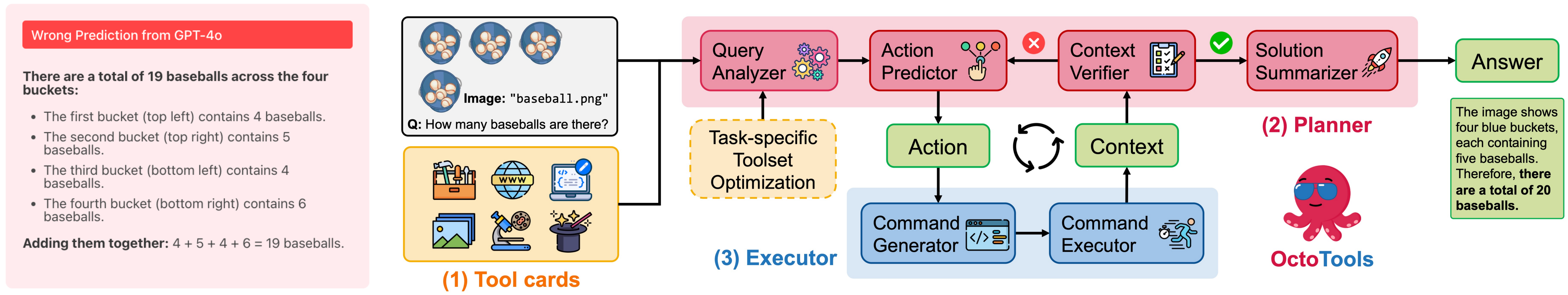


Image Captioner Tool

tool_description=`"A tool that generates captions for images."`

inputs=`{image, prompt}`

outputs=`{str: "The generated caption."}`

demos=`tool.execute(image="image.png", prompt="Describe the image.")`

user_metadata=`{"limitation": "May make mistakes in complex scenes & counting."}`

Object Detector Tool

tool_description=`"A tool that detects and saves objects in an image"`

inputs=`{image,labels,threshold}`

outputs=`{list: objects, scores, boxes}`

demos=`tool.execute(image="car.png", labels=["car"])`

user_metadata=`{"limitation": "May not detect objects accurately."}`

......

Step 0: Query Analyzer

Summary: determine the total number of baseballs in the image.

Required skills:
1. understanding and interpreting visual content.
2. ability to operate and execute commands using the tools.
3. evaluating tool outputs and making decisions.

Relevant tools:
`Image_Captioner_Tool`
`Object_Detector_Tool`

Additional considerations: Consider the limitations of each tool, e.g., potential inaccuracies in object detection in complex scenes or object detection. Verify results and be aware of the need for supplementary tools for precise counting.

Step 1: Action Predictor

Context: Image path: "`baseball.png`"
Sub-goal: Generate a description of the image to identify baseballs and relevant objects.
Tool name: `Image_Captioner_Tool`

Step 1: Command Generator

Analysis: The task requires using `Image Captioner Tool` to generate a description of an image to identify baseballs. The metadata specifies that it needs an image path and a prompt.
Command: `tool.execute(image="baseball.png", prompt="Describe this image.")`

Step 1: Command Executor

The image shows four blue buckets, each containing five baseballs. Each bucket has a handle on the side.

Step 1: Context Verifier

Completeness: Context provides a description of the image.
Inconsistencies: There are no inconsistencies.
Verification: Given limitations of `Image Captioner Tool` in counting, verify the count using `Object Detector Tool`.
Ambiguities: There are no ambiguities.
Conclusion: `CONTINUE`

Step 2: Action Predictor

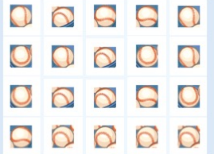
Context: Image path: "`baseball.png`"
Sub-goal: Detect and count the number of baseballs.
Tool name: `Object_Detector_Tool`

Step 2: Command Generator

Analysis: The task requires detecting baseballs using `Object Detector Tool`. It needs an image path and a 'baseball' label.
Command: `tool.execute(image="baseball.png", labels=["baseball"])`

Step 2: Command Executor

```
{
  "label": "baseball",
  "confidence score": 0.69,
  "box": [558, 48, 615, 107],
  "saved_path": "baseball_1.png"
},
...
```



Step 2: Context Verifier

Completeness: Context addresses the query using tools.
Inconsistencies: There are no inconsistencies in outputs.
Verification: Consistent results reduce need for verification.
Ambiguities: There are no ambiguities in the results.
Conclusion: `STOP`

Prediction from the OctoTool Framework

Planner Components | Executor Components | Final Answer

- Step 0: Initial Plan from Query Analyzer
- Step 1: Action from Action Predictor
- Step 1: Command from Command Generator
- Step 1: Result from Command Executor
- Step 1: Verification from Context Verifier
- Step 2: Action from Action Predictor
- Step 2: Command from Command Generator
- Step 2: Result from Command Executor
- Step 2: Verification from Context Verifier
- Full Trajectory in the Context
- Summary from Solution Summarizer
- Final Answer

Summary: The query aimed to determine the number of baseballs in an image. The process involved using two tools: an `Image_Captioner_Tool` to describe the image and an `Object_Detector_Tool` to count the baseballs.

Detailed Analysis

Step 1: `Image_Captioner_Tool`

- Result: The image contains four blue buckets, each with five baseballs, arranged in a grid pattern.

Step 2: `Object_Detector_Tool`

- Result: Detected 20 baseballs with varying confidence scores.

Key Findings

- The image contains a total of 20 baseballs, distributed evenly across four buckets.
- Each bucket contains five baseballs, as confirmed by both tools.

Answer to the Query

The image shows four blue buckets, each containing five baseballs. Therefore, **there are a total of 20 baseballs.**

Additional Insights

- The consistent results from both tools reinforce the accuracy of the analysis.
- The arrangement of the buckets and baseballs is clear and well-organized, aiding in accurate detection.

Wrong Prediction from GPT-4o

There are a total of 19 baseballs across the four buckets:

- The first bucket (top left) contains 4 baseballs.
- The second bucket (top right) contains 5 baseballs.
- The third bucket (bottom left) contains 4 baseballs.
- The fourth bucket (bottom right) contains 6 baseballs.

Adding them together: 4 + 5 + 4 + 6 = 19 baseballs.

https://octotools.github.io/#visualization

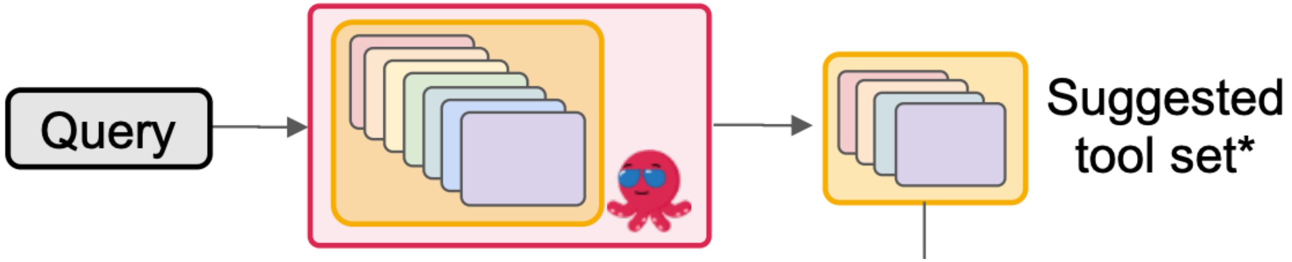
Task-Specific Toolset Optimization

Google_Search
Wikipedia_Knowledge_Searcher
Arxiv_Paper_Searcher
url_text_extractor

Python_Code_Generator

Image_Captioner
Text_Detector
Relevant_Patch_Zoomer
Advanced_Object_Detector

Generalist_Solution_Generator



16 Benchmarks

Datasets	Modality	Domain				
VQA 2.0	Vision	General	✓			
Hallusion-VD	Vision	General	✓			
AlgoPuzzleVQA	Vision	General	✓			✓
PuzzleVQA	Vision	General	✓			
Game of 24	Text	Mathematical		✓		✓
Omni-MATH	Text	Mathematical		✓	✓	
CLEVR-Math	Vision	Mathematical	✓	✓		
MathVista	Vision	Mathematical	✓	✓	✓	✓
GPQA	Text	Scientific			✓	✓
MMLU-Pro	Text	Scientific			✓	✓
SciFIBench	Vision	Scientific	✓		✓	
MedQA	Text	Medical			✓	
PathCLS	Vision	Medical	✓		✓	
PathVQA	Vision	Medical	✓		✓	✓
SLAKE	Vision	Medical	✓		✓	
GAIA-Text	Text	Agentic		✓	✓	✓

2 modalities
5 domains
4 reasoning types

- visual understanding
- numerical calculation
- knowledge retrieval
- multi-step reasoning

Setups

- Each sampled 200 examples
- Report average accuracy from three trials
- GPT-4o (2024-08-16)
- 10 steps, 300 seconds

Comparisons with Baselines and Agentic Frameworks

Datasets	Modality	Domain				
VQA 2.0	Vision	General	✓			
Hallusion-VD	Vision	General	✓			
AlgoPuzzleVQA	Vision	General	✓			✓
PuzzleVQA	Vision	General	✓			
Game of 24	Text	Mathematical		✓		✓
Omni-MATH	Text	Mathematical		✓	✓	
CLEVR-Math	Vision	Mathematical	✓	✓		
MathVista	Vision	Mathematical	✓	✓	✓	✓
GPQA	Text	Scientific			✓	✓
MMLU-Pro	Text	Scientific			✓	✓
SciFIBench	Vision	Scientific	✓		✓	
MedQA	Text	Medical			✓	
PathCLS	Vision	Medical	✓		✓	
PathVQA	Vision	Medical	✓		✓	✓
SLAKE	Vision	Medical	✓		✓	
GAIA-Text	Text	Agentic		✓	✓	✓

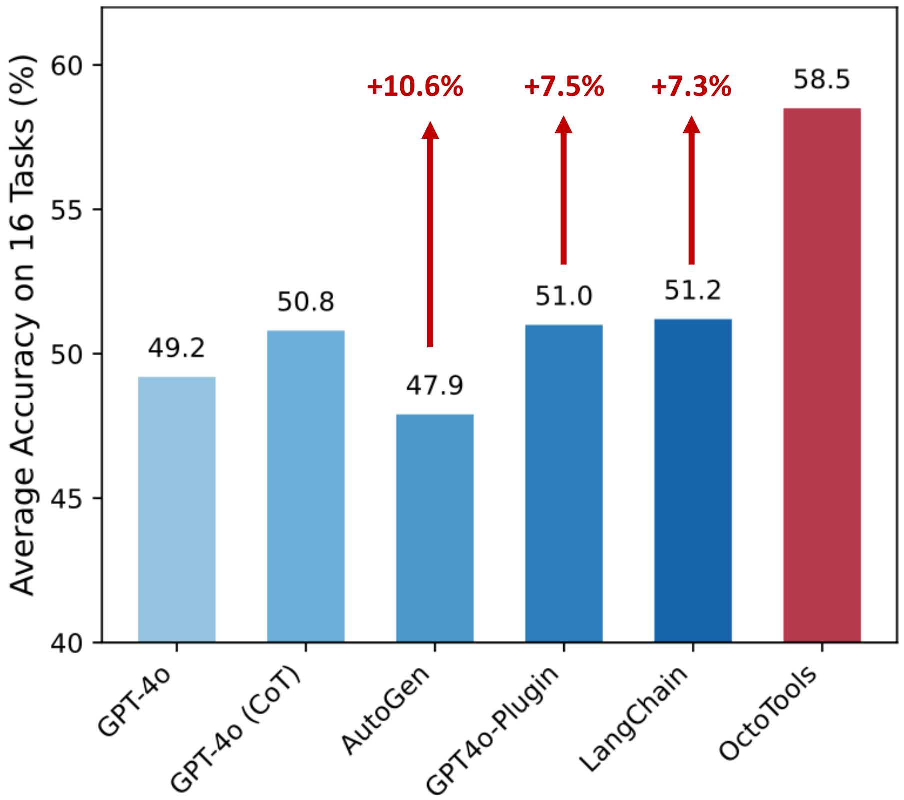


Figure 1. Performance comparison across 16 benchmarks. On average, our OctoTools system achieves an average accuracy gain of 9.3% over GPT-4o and 7.3% over LangChain.

Main Results

Datasets	Modality	Domain					0-shot	CoT	OctoTools (1)	OctoTools	Δ (0-shot)	Δ (CoT)
VQA 2.0	Vision	General	✓				50.3 ±1.0	48.7 ±0.3	47.2 ±0.8	**54.5** ±0.0	+4.2	+5.8
Hallusion-VD	Vision	General	✓				52.0 ±1.0	53.3 ±2.1	59.0 ±0.0	**63.3** ±2.9	+11.3	+10.0
AlgoPuzzleVQA	Vision	General	✓			✓	41.3 ±0.3	42.7 ±1.0	44.0 ±0.9	**48.7** ±0.3	+7.4	+6.0
PuzzleVQA	Vision	General	✓				52.2 ±1.0	54.0 ±1.3	59.3 ±0.8	**61.0** ±0.5	+8.8	+7.0
Game of 24	Text	Mathematical		✓		✓	22.2 ±2.5	33.3 ±1.5	37.8 ±3.3	**44.7** ±2.8	+22.5	+11.4
Omni-MATH	Text	Mathematical		✓	✓		27.0 ±0.0	29.3 ±1.3	30.2 ±0.6	**32.2** ±0.8	+5.2	+2.9
CLEVR-Math	Vision	Mathematical	✓	✓			64.5 ±3.0	75.2 ±1.5	68.8 ±0.8	**79.0** ±0.9	+14.5	+3.8
MathVista	Vision	Mathematical	✓	✓	✓	✓	59.3 ±0.8	59.5 ±1.5	63.0 ±1.3	**64.3** ±1.0	+5.0	+4.8
GPQA	Text	Scientific			✓	✓	53.7 ±1.9	52.3 ±2.0	53.7 ±2.5	**54.7** ±1.3	+1.0	+2.4
MMLU-Pro	Text	Scientific			✓	✓	71.7 ±0.3	70.3 ±0.6	71.5 ±1.3	**73.7** ±1.3	+2.0	+3.4
SciFIBench	Vision	Scientific	✓		✓		72.5 ±0.0	75.0 ±0.9	77.3 ±0.8	**78.3** ±0.6	+5.8	+3.3
MedQA	Text	Medical			✓		84.5 ±1.0	84.8 ±0.6	**92.8** ±0.6	91.5 ±1.8	+7.0	+6.7
PathCLS	Vision	Medical	✓		✓		36.0 ±0.9	37.5 ±1.8	37.0 ±1.8	**58.2** ±1.3	+22.2	+20.7
PathVQA	Vision	Medical	✓		✓	✓	32.0 ±1.8	27.8 ±1.8	43.5 ±2.6	**49.2** ±1.2	+17.2	+21.4
SLAKE	Vision	Medical	✓		✓		59.3 ±1.0	60.3 ±0.6	59.2 ±1.8	**63.8** ±1.4	+4.5	+3.5
GAIA-Text	Text	Agentic		✓	✓	✓	8.7 ±0.8	8.4 ±0.5	9.7 ±0.9	**18.4** ±1.2	+9.7	+10.0
Average (%)	-	-	-	-	-	-	49.2	50.8	53.4	**58.5**	**+9.3**	**+7.7**

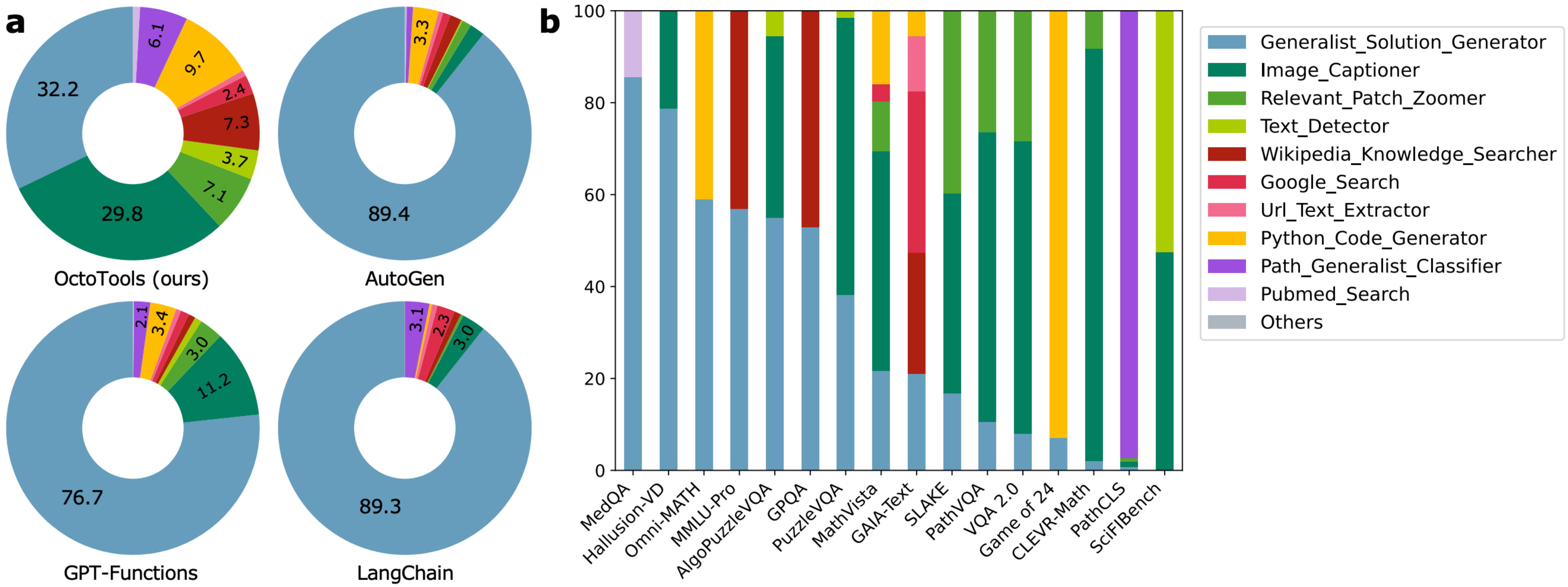
Comparisons with Other Agent Systems

Datasets	AutoGen	GPT4o-Plugin	LangChain	OctoTools
VQA 2.0	46.0 ±1.0	45.5 ±0.9	54.0 ±1.0	**54.5** ±0.0
Hallusion-VD	52.7 ±4.7	57.0 ±1.7	53.7 ±3.1	**63.3** ±2.9
AlgoPuzzleVQA	44.0 ±1.0	44.5 ±0.5	42.7 ±2.8	**48.7** ±0.3
Puzzle VQA	40.0 ±2.3	52.5 ±2.8	53.5 ±7.8	**61.0** ±0.5
Game of 24	24.2 ±2.4	34.5 ±2.3	18.3 ±4.1	**44.7** ±2.8
Omni-MATH	28.5 ±1.3	22.8 ±1.8	29.7 ±0.6	**32.2** ±0.8
CLEVR-Math	69.5 ±3.9	71.2 ±1.0	69.2 ±4.6	**79.0** ±0.9
MathVista	24.7 ±2.5	54.5 ±2.0	55.7 ±0.3	**64.3** ±1.0
GPQA	48.7 ±2.9	45.8 ±2.6	52.2 ±1.2	**54.7** ±1.3
MMLU-Pro	65.0 ±2.5	65.8 ±2.4	70.3 ±1.2	**73.7** ±1.3
SciFIBench	70.0 ±2.2	68.8 ±3.2	77.0 ±0.5	**78.3** ±0.6
MedQA	83.7 ±2.8	84.8 ±0.3	73.7 ±0.6	**91.5** ±1.8
PathCLS	58.0 ±1.3	58.2 ±0.6	56.3 ±1.3	**58.2** ±1.3
PathVQA	42.7 ±0.8	42.8 ±2.3	45.7 ±4.4	**49.2** ±1.2
SLAKE	62.2 ±1.8	59.7 ±1.9	59.3 ±0.8	**63.8** ±1.4
GAIA-Text	6.3 ±0.8	7.9 ±0.8	7.6 ±1.2	**18.4** ±1.2
Average (%)	47.9	51.0	51.2	**58.5**

Setups

- Same toolset along with metadata
- Each sampled 200 examples
- Report average accuracy from three trials
- GPT-4o (2024-08-16)
- 10 steps, 300 seconds

Tool Usage Distribution



Number of Maximal Allowed Steps

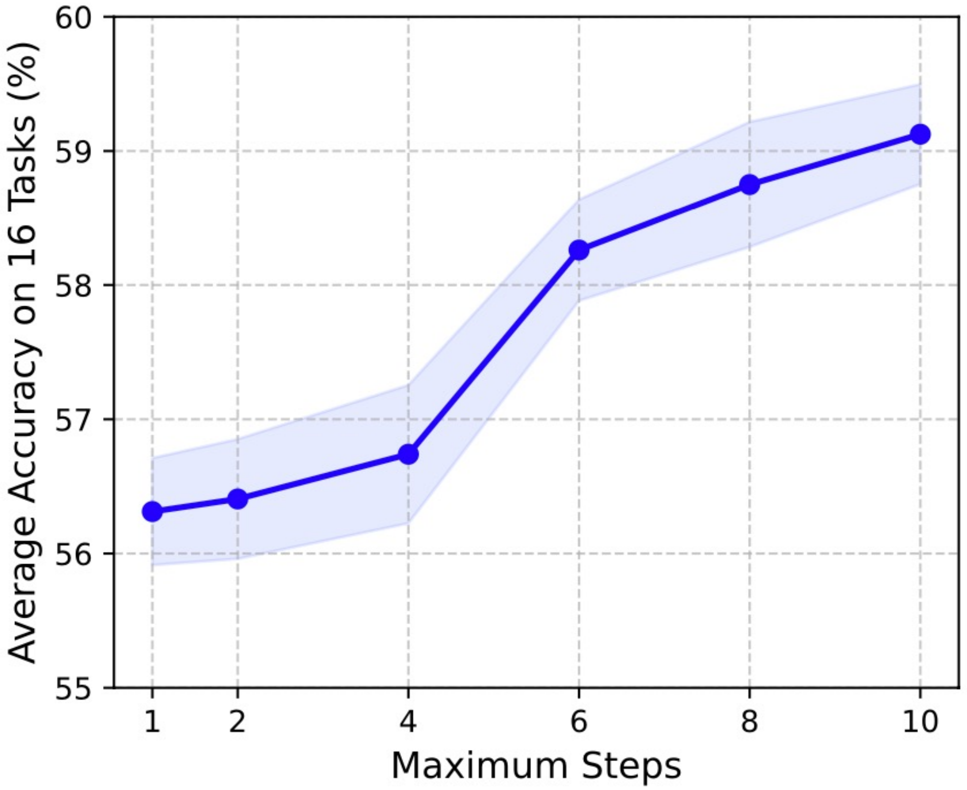


Figure 5. Average accuracy across 16 benchmarks with respect to maximum allowed reasoning steps in OctoTools. Performance generally improves with more steps and plateaus.

Performance tends to improve as the maximum number of steps increases, indicating the benefit of longer chains of multi-step reasoning.

- 1 to 4: modest
- 4-6: substantial gains
- Beyond 6: begin to plateau

Using a Weaker LLM

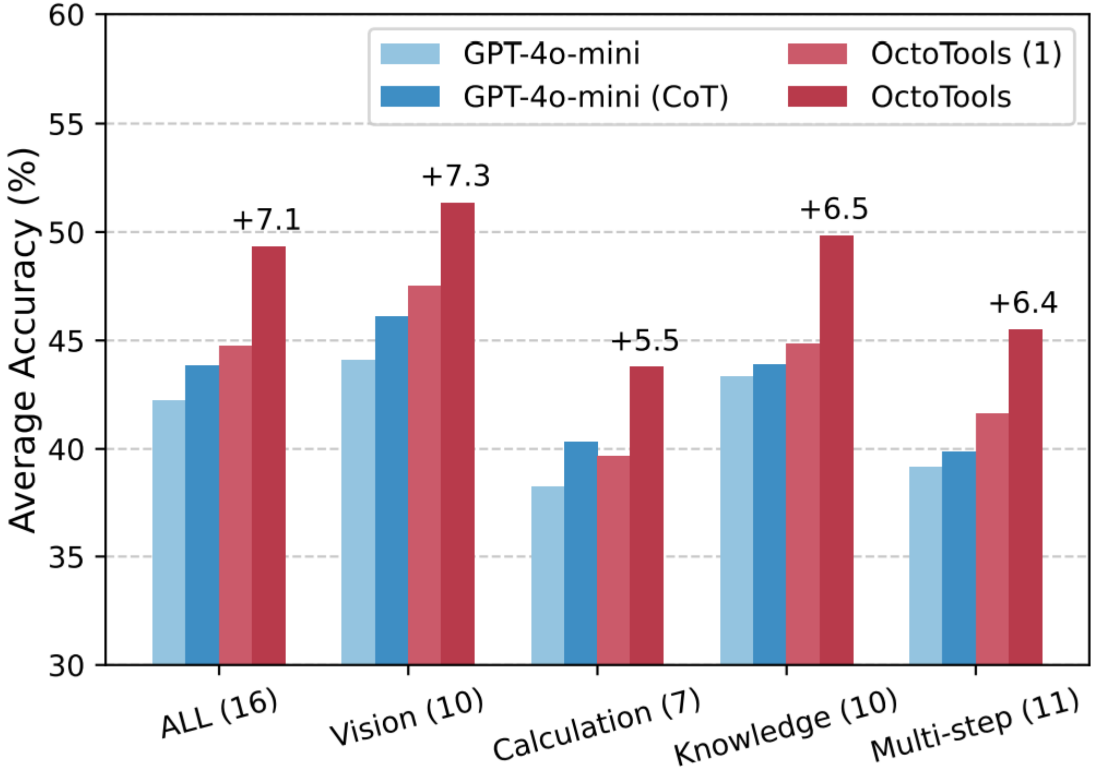
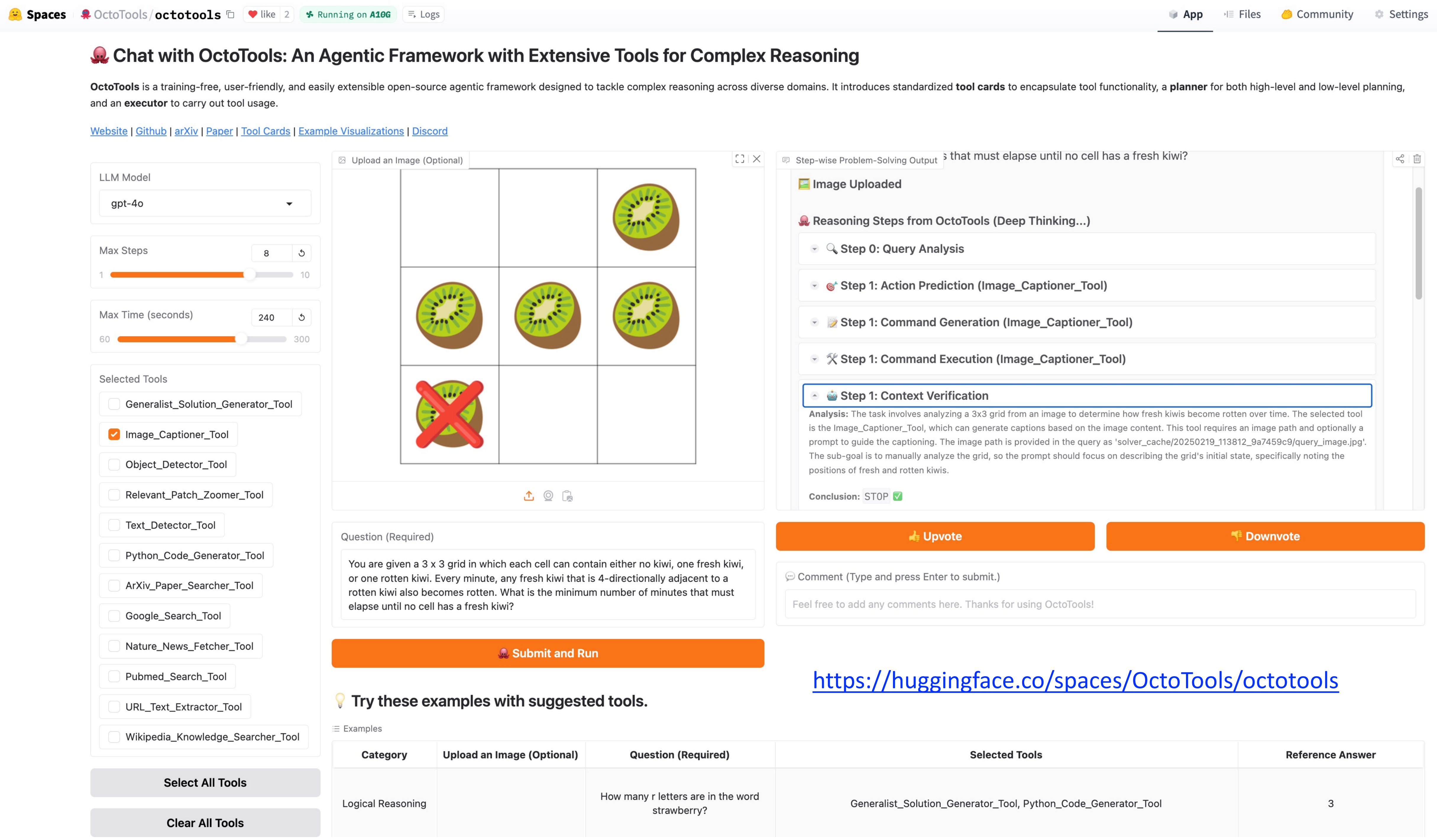


Figure 7. Performance of OctoTools on 16 tasks and various categories using a weaker LLM, GPT-4o-mini.

Datasets	**Modality**	**Domain**				
VQA 2.0	Vision	General	✓			
Hallusion-VD	Vision	General	✓			
AlgoPuzzleVQA	Vision	General	✓			✓
PuzzleVQA	Vision	General	✓			
Game of 24	Text	Mathematical		✓		✓
Omni-MATH	Text	Mathematical		✓	✓	
CLEVR-Math	Vision	Mathematical	✓	✓		
MathVista	Vision	Mathematical	✓	✓	✓	✓
GPQA	Text	Scientific			✓	✓
MMLU-Pro	Text	Scientific			✓	✓
SciFIBench	Vision	Scientific	✓		✓	
MedQA	Text	Medical			✓	
PathCLS	Vision	Medical	✓		✓	
PathVQA	Vision	Medical	✓		✓	✓
SLAKE	Vision	Medical	✓		✓	
GAIA-Text	Text	Agentic		✓	✓	✓



https://huggingface.co/spaces/OctoTools/octotools

Using AI is financially and environmentally expensive

Model: GPT-4

J

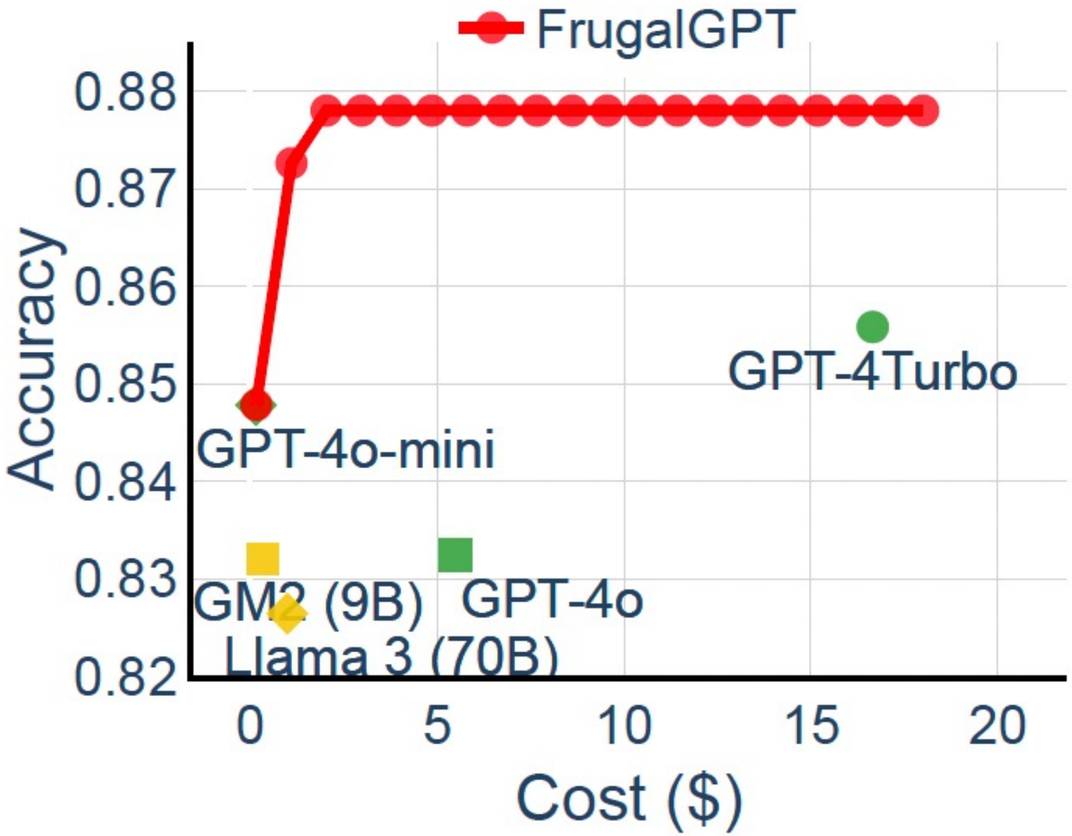
How much energy does it cost to generate each token by you?

As of my knowledge cutoff in September 2021, OpenAI had not publicly provided specific numbers on the energy consumption of GPT-3 or any of its predecessor models per token generated.

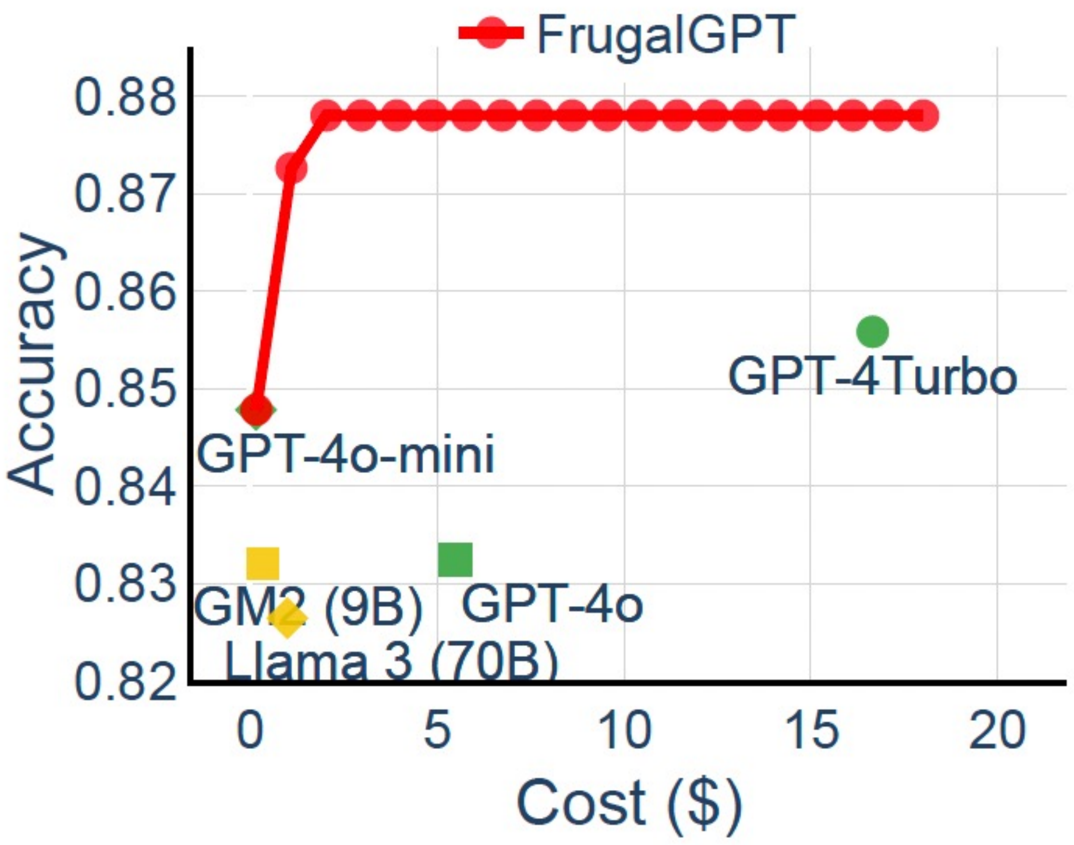
It's important to note that the energy cost of running these models depends on several factors:

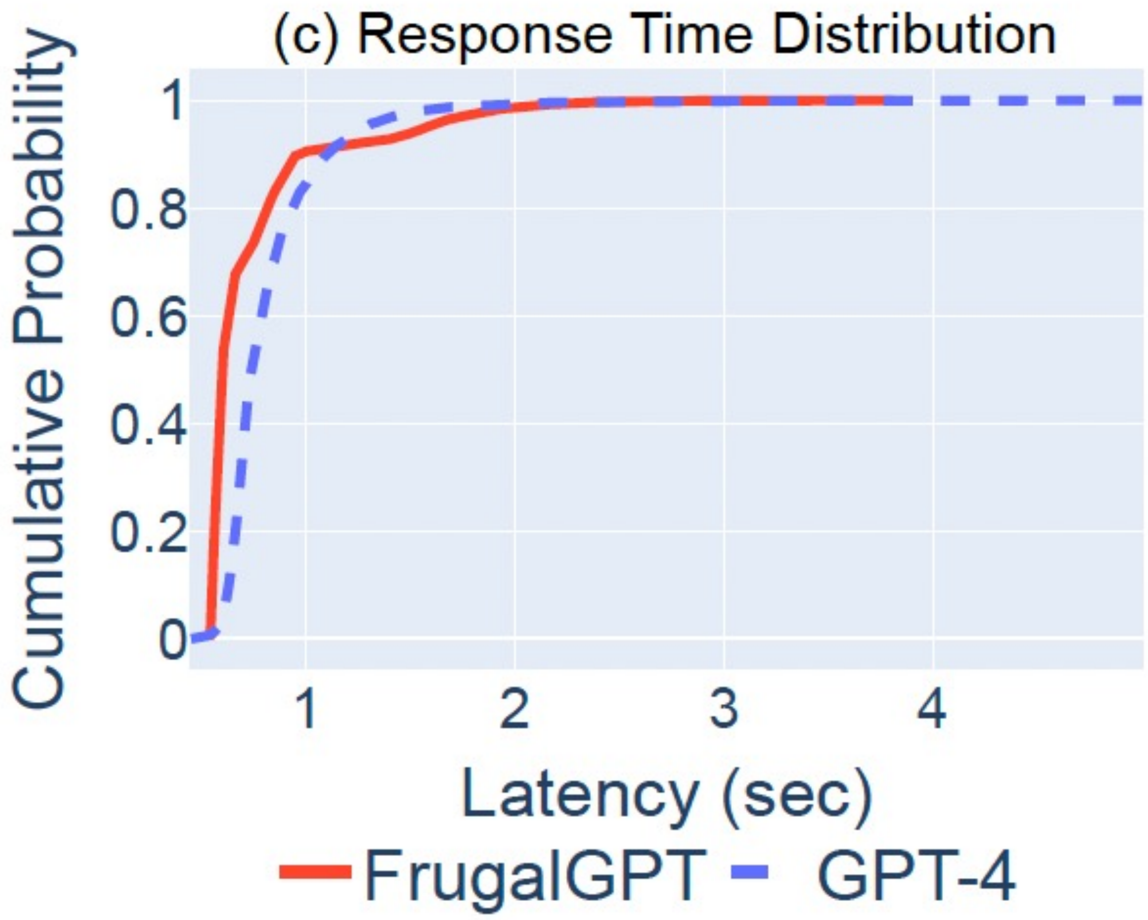
1. **Hardware specifications**: More powerful hardware often uses more energy, but it can also process tasks more efficiently.
2. **Model size and complexity**: Larger models like GPT-3 require more computational resources, and thus more energy, to run.
3. **Server efficiency and cooling**: These models run in data centers, where cooling systems, server efficiency, and even the source of electricity can significantly affect overall energy consumption.
4. **Model optimization and fine-tuning**: The more optimized the model is, the less computation (and therefore energy) it needs to generate a response.

FrugalGPT improves over GPT-4o at a fraction of cost

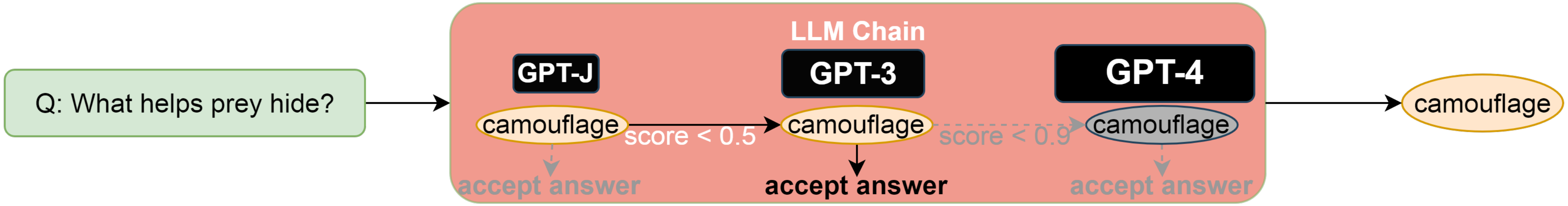


FrugalGPT improves over GPT-4o at a fraction of cost



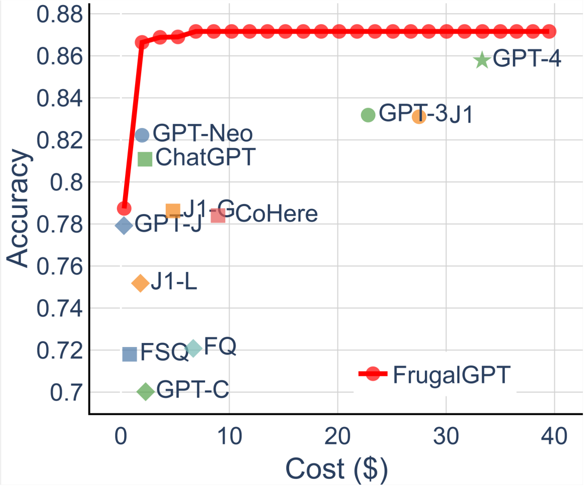


LLM cascade

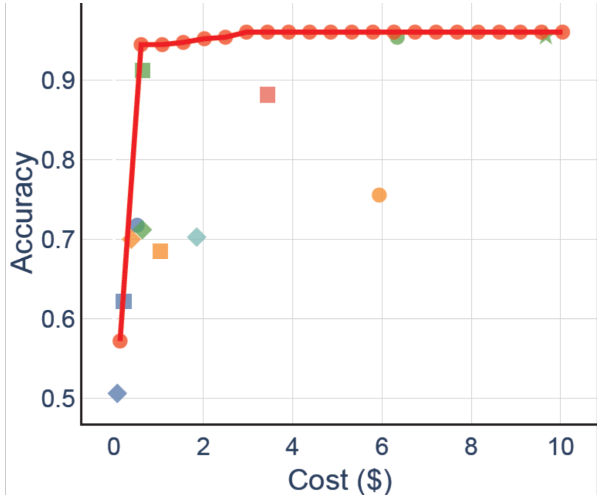


Adaptively select which LLMs to use

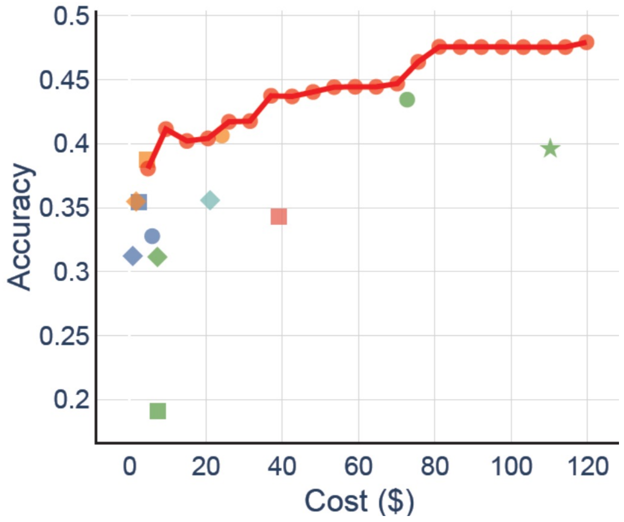
FrugalGPT optimizes performance and cost tradeoffs



HEADLINES



OVERRULING



COQA

FrugalGPT optimizes performance and cost tradeoffs

Table 2: Cost (USD) savings by FrugalGPT to match the best individual LLM's performance.

Dataset	Best invidual LLM	Cost to reach the same accuracy		Cost Savings
		Best individual LLM	FrugalGPT	
HEADLINES	GPT-4	33.1	0.6	98.3%
OVERULLING	GPT-4	9.7	2.6	73.3%
COQA	GPT-3	72.5	29.6	59.2%
AGNEWS	GPT-4	64.6	15.9	75.4%
SCIQ	GPT-3	132.4	63.1	52.3%